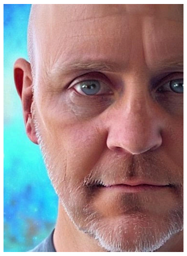
Network Basics Survival Guide



Introduction - note from the author.

-Will McCullen

Welcome to my version of network basics. There are many out there. My hope is this is a version that helps make the topic a little easier.

What this document is not:

Let me be clear. This document is not an in depth resource on networking. There are excellent resources that represent good sources of learning networking. The CompTIA Network+ classes, for example, provide a strong foundation of knowledge that someone needs to really succeed in this industry. Other resources go into much more detail and depth than I explore here.

What this document is:

This document was developed as a kickoff into the Introduction to Wireshark class. The basic premise is to refresh the memory of students that have taken the Network + classes, and cover some of the most basic concepts that will be required for understanding Wireshark and the information it displays for analysis. For those that are not familiar with Wireshark, it is a fantastic tool. It is the industry standard tool for looking at network information as it flows across the medium, and allows you to analyze and make sense of the packets. It is brilliant through its use of filtering, coloring, and graphing to make that illusive needle glow in the haystack. It can help you find small details in a mass of packet information. However:

–It is very hard to use Wireshark, if you are not comfortable with the basics.

That is where this document comes in. This document is meant to trigger memories of what you learned previously, and hopefully give a comfortable, and maybe even fun, context of why the information you learned is so important. If you have stumbled across this document and you are currently learning networking, then hopefully this helps as well. The goal is to simplify a little of what can ordinarily be dry reading. Let's see how I do.

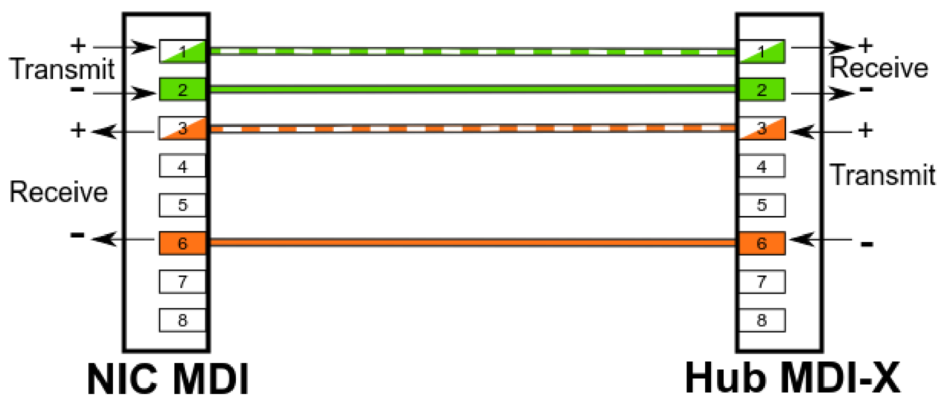
Contents:

Getting started, how in the world does this stuff work?

Putting data on the network: - Layer 1, the physical layer.

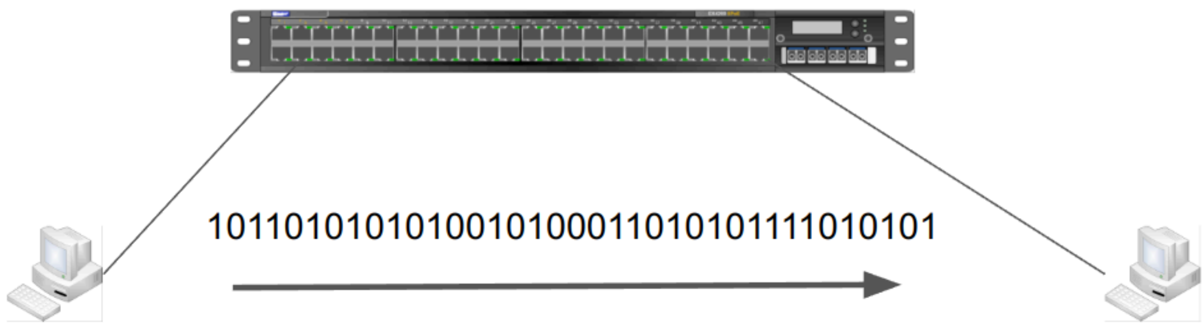
The computer is an awesome box. It does all sorts of incredible things, which is fine in its own right, but things get really interesting and powerful when we start to pass data from the computer to some other computer across a network. Chances are your computer is probably connected over WiFi. That has become more and more the standard these days, as it is the easy way to do things at home, or almost anywhere, since you do not have to run wire. However, if you are at work then chances are pretty good that your computer is plugged into the network via an ethernet cable. This is the best place to start as ethernet has been around for a while, and gives a clearer picture of how networks, and the principles that have grown with them, develop.

Have you put much thought into how data gets from point A to point B? I hope it has developed some curiosity as that is basically what this whole document is about. The data that we pass is essentially, like all data, just a bunch of bits. Ones and zeros travel across at incredible speeds. So, the concept of networking can be thought of as simply creating rules that bring some organization to how that can be done in a predictable fashion where it can be understood on the other end. When you pass information across a wire, you are simply turning an electric circuit on and off.

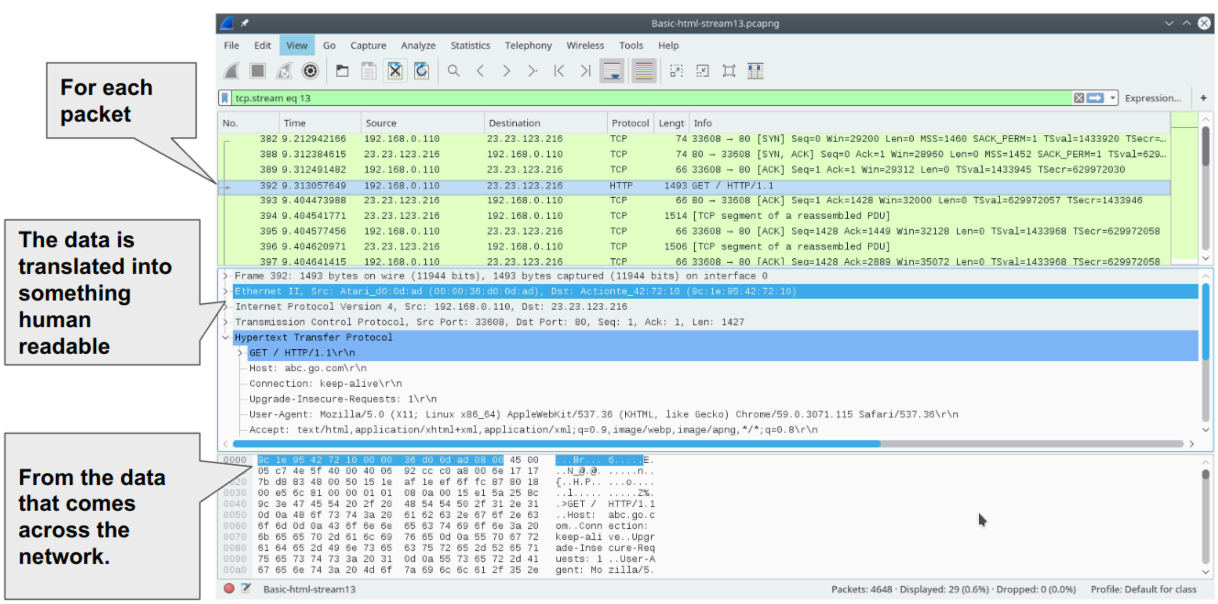


Just like a switch connected to a battery and a light. When you turn the switch on and off it is the equivalent of sending an “on” or an “off”. Flip that switch really fast and you are transferring bits. In copper unshielded twisted pair (a type of ethernet cable), you have a circuit for transmit and one for receive so both can take place at the same time, just like when you talk on the phone. You can talk and hear at the same time. That is known as “Full Duplex”. “Half Duplex is like a radio. You can only talk when pressing the button. You can only hear when you're not. When you use fiber, it isn’t a circuit, it is turning a laser light on and off very quickly. It is the same concept though, you have a fiber pair. One fiber for transmit and the other for receive.

Okay, now we have the basic concept of how we are sending and receiving bits. Wonderful. The next step is to make sense out of it. If you have taken network classes, do you remember the Open Systems Interconnection (OSI) model? The very first layer is the physical layer. When one host on a network starts sending data to another host on a network there are a series of bits called the preamble. That is what lets the other host know that data is starting to come in. You will probably never see that information because it is hidden by the Network Interface Card (NIC), so we will not be able to see it in Wireshark.



Once the data starts to stream in, the next task is to make sense of it. It is simply a matter of both sides knowing what to expect from the data. In other words, what piece of information will be first, second, third, etc. The agreed upon information structure, or rules, are what are known as protocols. When that information streams in, it is simply translated into bytes, which as humans, can be really ugly to deal with. That is where awesome programs like Wireshark come in. Wireshark is a network traffic analyzer, it takes the raw data and presents it in a way that is much easier to read, understand, and filter.

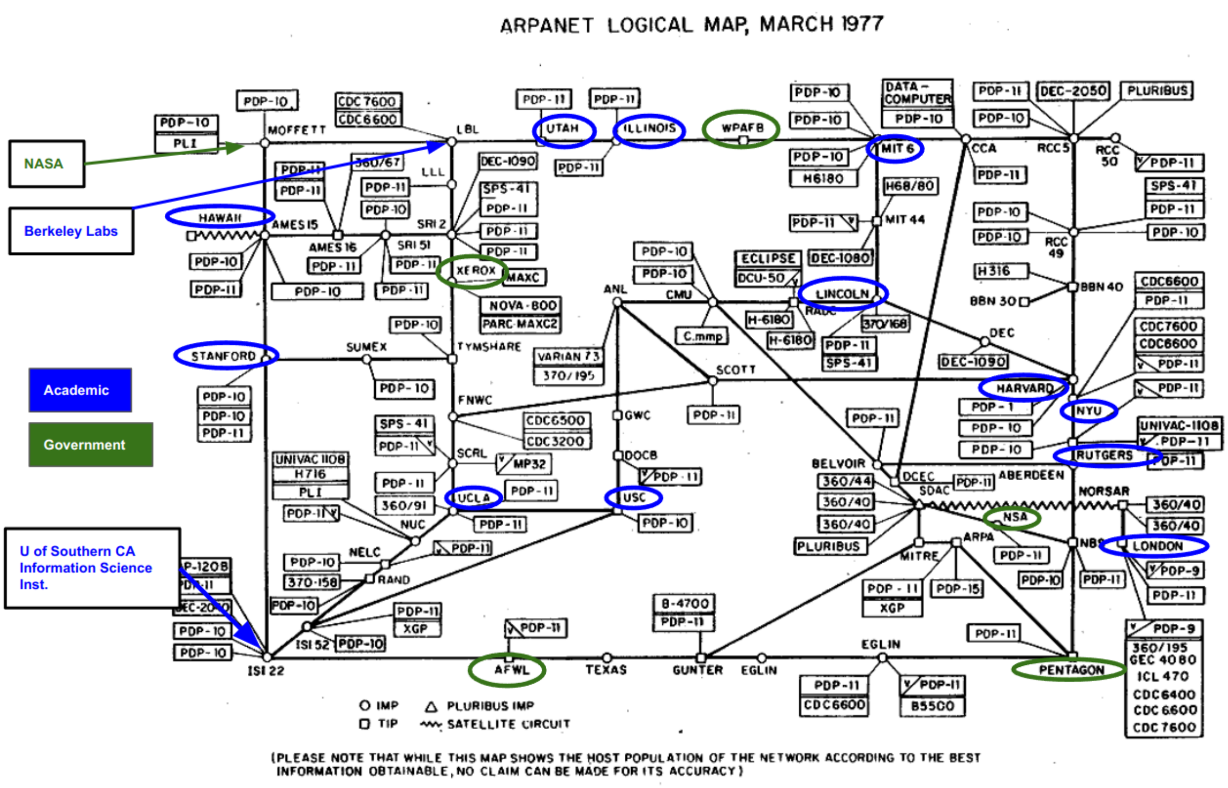


Sidetrack:
When discussing bits, you may come across the terms “byte” and “octet”. An “octet” is 8 bits. Historically a byte represents the number of bits to represent a single character of text. That varied in older systems, but now the standard is 8 bits and is used synonymously with “octet”. Fun fact: half a byte is a “nibble”.

Part 1

First, a little bit of history for context.

When folks were first developing networking protocols, the requirements were minimal. It started as a solution that could pass data from one large system to another across the country. It was a direct connection that did not need protocol complexity. Early solutions were based upon creating virtual circuits. Think of virtual circuits as having to establish with the phone company a direct telephone line where you can’t dial anyone else. The phone line never hangs up, it is always connected. You would have to create a separate line to everyone you wanted to talk to. When the systems were few, this solution had some distinct advantages. However, it did not scale well. The solution to overcome individual circuits was the concept of packet switching. Packets conceptually worked more like postal mail and only required one line to the shared network. Each packet (letter) would have a source and destination address. The packets can be sent/received, sorted, and can take different routes along the way till it gets to the final post office, and then to your house. Once they all get to your house you can take all the paper and put it back together to get the full conversation. The packet based protocol that we use today to communicate across networks is Transmission Control Protocol/Internet Protocol (TCP/IP).



The concept of packet switching networks really started gaining a research foothold in 1970, and by 1977 as you can see had quite a few institutions involved. Probably the most dominant forces were the computing research resources being developed at major universities. (Blue)

However, there was a strong interest by the government/military (Green), and they developed their own network infrastructures as well. In 1980 both the Advanced Research Projects Agency Network (ARPANET) and Department of Defense (DOD) DARPA research projects based their standard on TCP/IP. Over time it was developed to improve the whole packet switching networking solution, and became the protocol of choice. There had been many influences into the creation of the TCP/IP protocol, so the protocol itself was not under licensing restrictions like the other protocols that were trying to gain the most dominant position in industry. TCP/IP being very efficient, used by the growing Internet, and free on top of that, found itself becoming the most dominant protocol in the world.

At the same time that TCP/IP was gaining traction in the DOD and Academic community, the main demand for businesses was to be able to have their new environment of personal workstation computers all connected to the same printer and file server, to reach the businesses resources. Networking for the business started with the local network as the focus. Tech companies grew quickly. IBM/Microsoft, Apple and Novell grew, taking advantage of the need for lower cost systems. Along with that, as their systems grew, they all developed their own method of networking, their own proprietary licensed protocols. These solutions quickly grew to more complex requirements. When this early networking was developing, companies were all racing to develop their own solutions. Their hope was, if their solution with their own protocols was able to take the dominant spot in the marketplace, their solution would then hopefully become the computing standard. It created a protocol war[1]. The influential network protocols included IBM (SNA), Digital Equipment Corporation (DEC Net), Apple (AppleTalk), Novell Networks (IPX/SPX), and IBM (NetBIOS). However, the protocol that won out was free, developed by the government, and was used to provide communication across the global community. Once the world wide web came on the scene with the development of web browsers, there was an application that had a unified demand across all companies. That protocol was of course TCP/IP.

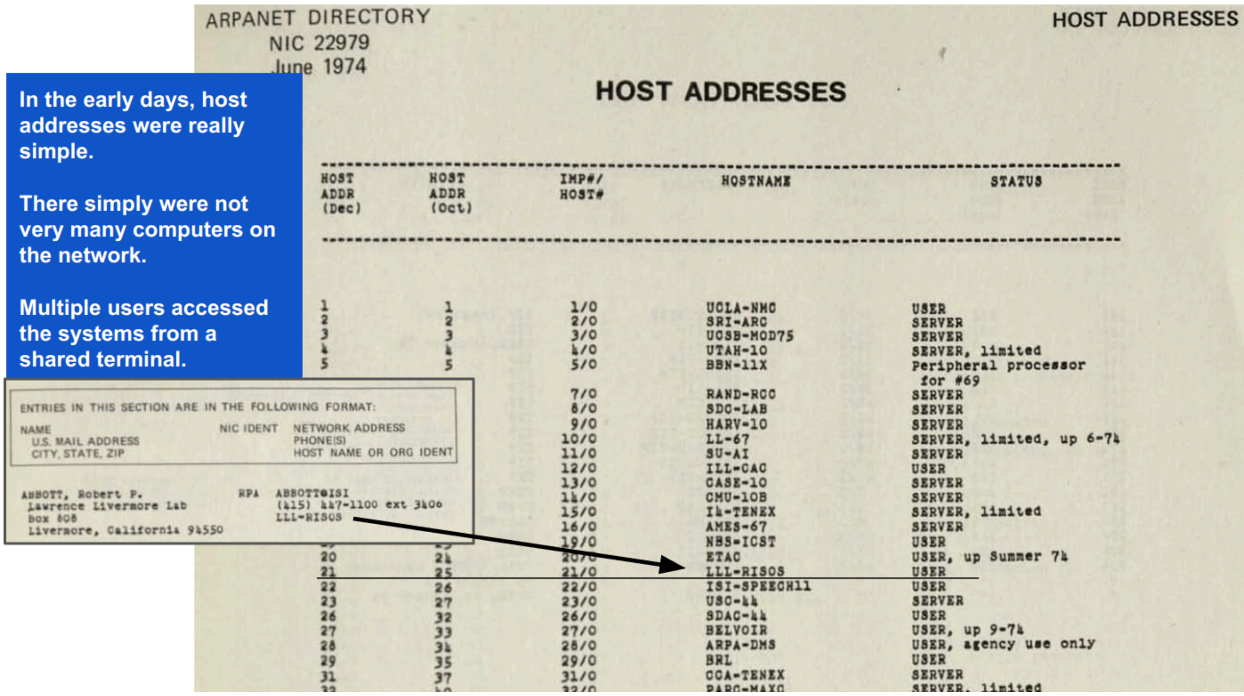
It is important to understand that adoption of a protocol was not thoroughly thought out at the beginning. It was something that developed over time and continues to develop to solve a great many varying applications.

Sidetrack:

To give some perspective, all of this grew from folks simply trying to get things done. It is easy to assume as you learn computing, that all this complexity was all well thought out along the way. The truth is, all of this was really developed by folks in the right place at the right time, cooperating with each other to achieve a goal. The original computer hackers if you will.

The first addresses in the ARPANET were pretty basic. There simply were not many computers on the network. This is from one of the ARPANET directories:

[1] "Protocol Wars - Wikipedia." https://en.wikipedia.org/wiki/Protocol_Wars. Accessed 2 Jan. 2023.



When engineers were first getting started they needed a solid way to compare notes and put out standards that others could build off of. These notes became “Request for Comments” (RFC). Some of the early ones demonstrate just how much the process was simply a matter of figuring it all out as they went: from RFC 33[2]

> *The elements of the name space are called sockets. A socket forms one end of a connection, and a connection is fully specified by a pair of sockets. A socket is specified by the concatenation of three numbers:*
>
> *(a) a user number (24 bits)*
> *(b) a HOST number (8 bits)*
> *(c) AEN (8 bits)*
>
> *A typical socket is illustrated in Figure 3.*
>
> *Each HOST is assigned all sockets in the name space which have field (b) equal to the HOST's own identification.*
>
> *A socket is either a receive socket or a send socket, and is so marked by the lower-order bit of the AEN (0 = receive, 1 = send). The other seven bits of the AEN simply provide a sizable population of sockets for each used number at each HOST. (AEN stands for "another eight-bit number")*

[2] "rfc33.txt - » RFC Editor." https://www.rfc-editor.org/rfc/rfc33.txt. Accessed 1 Jan. 2023.

“AEN stands for “another eight-bit number”. Really? Can’t you just see them sitting around discussing.

> *“Well, we need to have some bits set aside so we can do some of this other stuff we are thinking about.”*
> *“Okay, let's set aside an even 8 bits. What shall we call it?”*
> *“I don’t know. Let’s just call it an AEN till we figure it out”*

Moral of the story, as you start to see how things work, understand that some of the complexity grew because of how it developed over time, and not that it was clearly thought out at the beginning.

If you are really curious about the history and would like to see some of the early dialog, there were also IENs, or Internet Experiment Notes. You can find them on the same server as many of the RFCs. https://www.rfc-editor.org/ien/ien-index.html Fun fact, TCP used to be the only protocol. It wasn’t until a later version that it became TCP/IP. If you read IEN #2 you can catch a statement by Jon Postel recommending the creation of IP.

> *“We are screwing up in our design of internet protocols by violating the principle of layering. Specifically we are trying to use TCP to do two things: serve as a host level end to end protocol, and to serve as an internet packaging and routing protocol. These two things should be provided in a layered and modular way. I suggest that a new distinct **internetwork protocol** is needed, and that TCP be used strictly as a host level end to end protocol.”*[3]

And, thus…IP was born.

The technology that became “the networking solution” had to be based upon the concept of packet based switching. There was a huge need for a unified protocol, one that would be the glue for interconnectivity. This is what led to the drafting of the Open Systems Interconnection model (OSI model), and TCP/IP. The world was developing systems quickly. Technology was developing quickly, and everyone needed systems to communicate. The US academic community responded to this by developing Internet Protocol (IP). Its biggest advantage was that it was a working protocol that developed in a very dynamic open manner. Changes were communicated through a process called Request For Comments (RFC)[4]. Everyone who participated in the ARPANET was joined to this process for updates and changes as it grew. As the protocol developed, new problems would arise. The participants, programmers, and

[3] "IEN # 2 - » RFC Editor." https://www.rfc-editor.org/ien/ien2.txt. Accessed 23 Jul. 2023.
[4] "Request for Comments - Wikipedia." https://en.wikipedia.org/wiki/Request_for_Comments. Accessed 2 Jan. 2023.

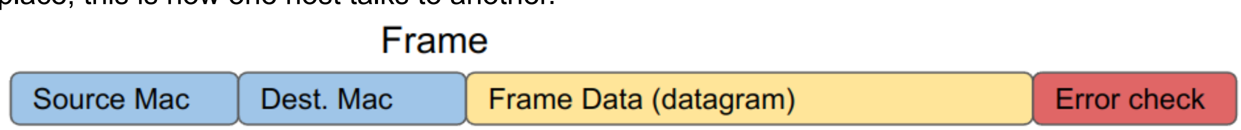
professionals leading the project, would regularly communicate solutions. Those solutions would then be coded and implemented amongst the community to become the rules that would have to be followed to make it work across the network, and then become official Comments. It was dynamic. It solved problems as they developed and was constantly changing. The most important element was that it allowed for communication to meet the current needs. Everyone needed the solutions right away.

However, in Europe and amongst international standards organizations, there was a very different approach. Their view was they needed, through internationally honored engineering committees, a solid agreed upon process to design the universal protocol. They wanted a universal protocol that was governed and agreed upon by their organization, the International Organization for Standardization (ISO). For many of these engineering bodies Internet Protocol (IP) was not even an option for universal adoption. IP was largely influenced by the US academic community, it was very dynamic, quickly changing and considered largely experimental. The frustration for the OSI efforts was that while they were developing their plan for what a universal protocol **should be**, many of the world's institutions were already actively using IP by 1980. That was when the OSI draft for what an OSI protocol **should be** was first published. I say "should be", because remember, the draft just represented a plan. No OSI protocol had been programmed yet. By the late 80s, adoption won out and IP solidified its way to becoming the world's networking protocol. As a result, the OSI model got relegated to only being a reference model, one that TCP/IP didn't really quite follow. No strict OSI overall unified protocol was ever developed. The original intention of the committees and engineering groups never really came to fruition, except to be mandated as conceptual content in curriculum and certifications across the world. It was a model for what all concepts in a protocol should include, but since IP took dominance, was free, and remained in active development, the protocol wars stopped. It was clear that no company was going to be able to create and capitalize on a proprietary protocol that everyone would have to adopt.[5]

Frame: The basic building block - Layer 2, the data link layer.

Once you start sending electrical, or light, or radio impulses that become data bits, then some structure is needed. How should that data be arranged to get to, and mean something at the other end? This is where protocols come into play. They are the rules that identify and govern how to interpret the bits coming through the network.

The frame is the basic building block. For your local ethernet network, where data links take place, this is how one host talks to another.



[5] "OSI: The Internet That Wasn't - IEEE Spectrum." 29 Jul. 2013, https://spectrum.ieee.org/osi-the-internet-that-wasnt. Accessed 8 Jul. 2023.

Every network interface that uses Ethernet has a **Medium Access Control (MAC) address**. A MAC address is six bytes in length. The first three bytes is what is called the **Organizational Unique Identifier (OUI)** and usually is a number assigned by the Institute of Electrical and Electronics Engineers (IEEE) to denote the manufacturer of the **Network Interface Card(NIC).**

MAC address 08:2E:5F:00:45:A6
08:2E:5F = HewlettP - Hewlett Packard

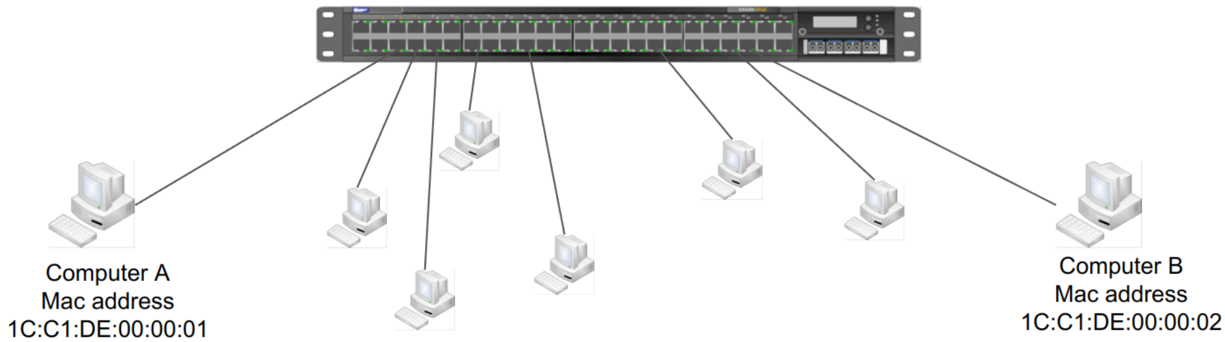
As manufacturers make equipment, they will use their OUI and use the other three bytes to count off each device they make. This system was created to make every interface unique. If you have two of the same MAC addresses on the same network…you can imagine how that goofs up which host actually gets the packet. It is the basic element of how computers can send a packet to a switch and recognize which host to send it to. At the end of a frame there is some error checking. Error checking is simply a math algorithm that when any bit is lost or altered in transmission, it can determine that the data did not arrive intact.

Sidetrack:
MAC addresses are used in all sorts of ways. Many times they are used for individual identification. Once a computer is identified in a unique way, it can be given special privileges, or blocked. It can also be tracked. First hint is in the OUI. There are lots of OUI lookup tools. Wireshark has a pretty decent one at: https://www.wireshark.org/tools/oui-lookup.html. If you have a MAC address, or even just the first three bytes you can enter it into the search box and perform a lookup. On that same page there is a link to the list Wireshark uses from the IEEE to do lookups in Wireshark. https://www.wireshark.org/download/automated/data/manuf Did you know there have been networking companies called Gandalf and Gothom? There has also been Fast, ClickTV, StarGate, CloudNet, PineTech and Stealth.

I bring this up because even though the MAC address is commonly used for a means of identification, it can also be changed, or spoofed. When you start to use Wireshark, this can be very useful, as it provides an easy method to identify your own machine in a capture. When you start looking at the data, it can be very helpful to have your address stand out as an Atari computer and then be able to think…oh, that is not what I am looking for. That is my computer.

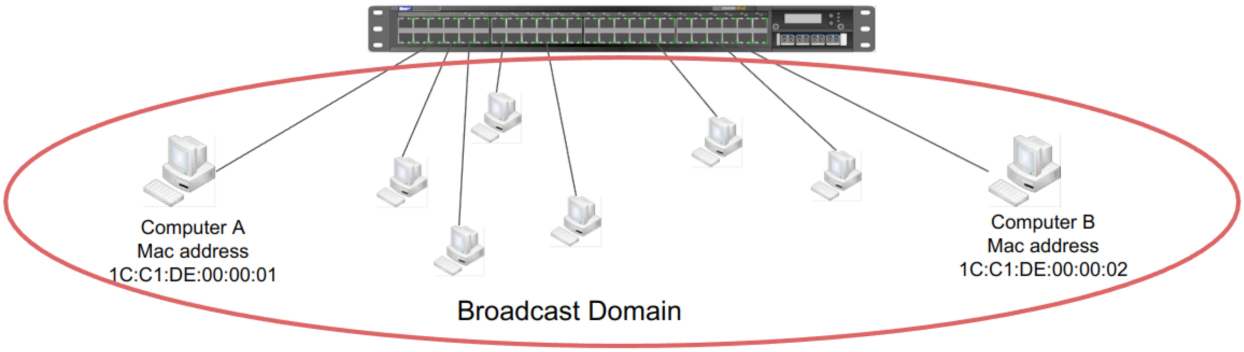
Just because you see a MAC address, it might be a clue, but it is not something that you can rely upon.

So let's think about this. The goal is to send data. We now have some basic rules that use the concept of addresses. That is great but how can they be sure to get where they need to go? In the early days of Ethernet, all computers hooked into a hub.



When data started to flow, everyone could see the data. The hub would duplicate the bits as they came into every computer that was connected. When a computer wanted to send information, it would first listen. If nobody was talking, then it would send the data starting with the destination MAC address. The computer with the destination MAC address would then be able to take the packet knowing that data was meant for it. The only problem was that if two computers tried to talk at the same time, the data would get intermixed and garbled. This is what is known as a **collision**. The solution to that problem was that if a computer started to get data coming in while it was sending, it would immediately stop sending and send a special signal denoting that there is a collision. If you think about that a little, it is kind of like everyone trying to have separate conversations on a conference call. The more people you get trying to talk, the harder it becomes. You quickly realize you need to be able to keep conversations separate, that is where switches come in.

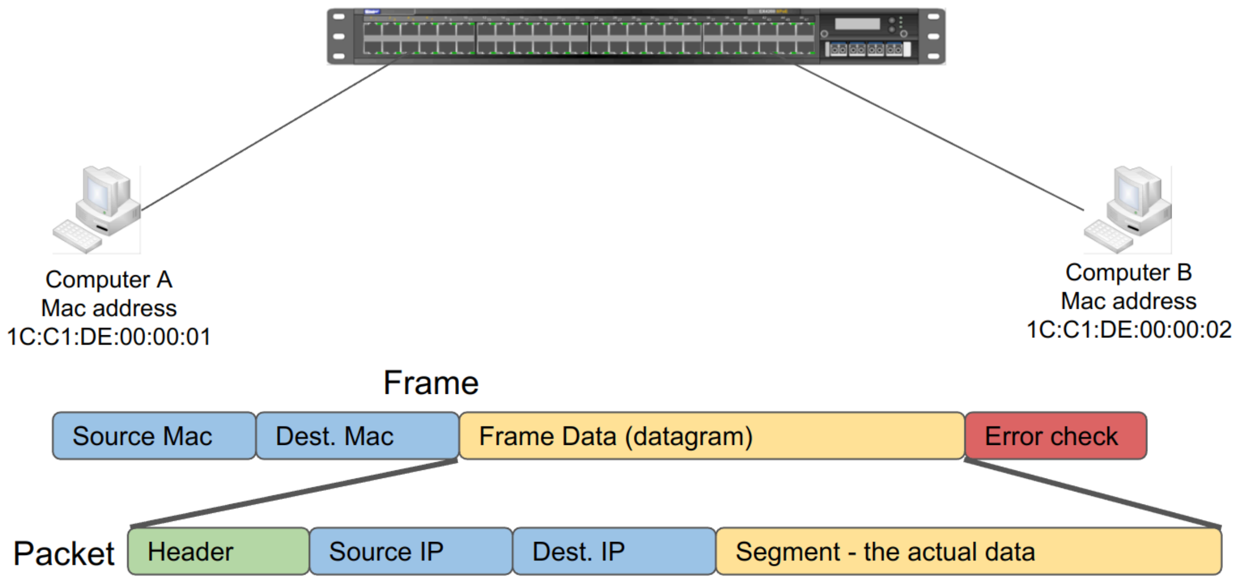
Switches basically solved the collision situation. If you look at the graphic where all the computers are plugged in, the device has multiple ports. For a switch, it developed a table called a **MAC address table**. That table keeps track of what MAC address is seen on which port. Switches also use a technique called "**Store and Forward"**. In other words, instead of sending bits on immediately as they stream in, the switch waits until it gets the whole frame from one computer before sending it on to the next. Once it has that frame, it checks its table, and then only sends the packet out to the port where it knows the destination address is. In other words, none of the other computers see the packet. They had no concept that the conversation was taking place between the two computers. Switches changed the network from being one big conference room to individual phone calls. Switches with the "Store and Forward" capability could simultaneously handle multiple conversations at the same time. This made a HUGE difference in network performance. This worked great, except there were some situations where a host on the network would need to contact everyone on the network at the same time. Built into the Ethernet protocol is the concept of a **broadcast**.



In Ethernet, when you send a packet with the destination MAC address of **FF:FF:FF:FF:FF:FF**, the switch will send that packet to everyone in that broadcast domain. This is a major concept to get used to. There is a lot that relies on this concept. This becomes especially important for communications with IP and the whole security concept of network isolation.

Packet: From network to network - Layer 3, the network layer.

Layer 2 has both source and destination addresses. It helped to get traffic from one source to another. At first glance it would seem that is all you need, but I would like you to remember the history. There were many networks when this was all starting up. Many of them were very different from one another. Ethernet is just one standard, and still is not the only standard we have today. It is simply the most common. Early on there was Token Ring, ARCnet, LocalTalk and Point to Point Protocol(PPP) for your local network. Backbone equipment still uses Asynchronous transfer mode (ATM), Fiber Distributed Data Interface (FDDI) and even the elderly Frame Relay, even though things are moving to Multiprotocol Label Switching (MPLS). Are all those acronyms important? Not for this discussion, except to illustrate that there are very different Layer 2 (ish) protocols out there. The original problem that the IT world was facing was how to be able to communicate with other systems. That is where Layer 3 comes into play. MAC addresses are the source and destination for the local network, IP addresses become the addresses that allow communication to travel in and beyond the local network. To do that, IP packets are then put into Ethernet frames.



As data goes across the Internet, it will travel from network to network. The frame portion (layer 2 information) will constantly change, the packet however (layer 3) will stay the same from the source to the destination.

Okay, now you might be thinking. Two addresses, MAC and IP? How does that work? If you weren't thinking that…well, you may be now.

Remember, MAC addresses are needed on the local network, but IP addresses are what all the applications use. When you are using applications like browsers, terminals, remote software, you are always using IP. If you reference something with what is known as a **Fully Qualified Domain Name(FQDN)** like "www.somewhere.com", you are also using IP. Yet, anything you send to an IP address has to travel in a frame that uses MAC addresses. This is where one of the more important concepts of networking comes into play.

On a local network, when a computer with an IP address wants to connect with another computer with an IP address **on the same network**, it needs a method to find out which computer's MAC address goes with that IP. It does that by using a broadcast address. It is not too elegant. The host basically sends to the MAC address FF:FF:FF:FF:FF:FF (broadcast):

"**HEY!** Who has this IP address? Please tell me at - source MAC address and source IP."

Notice what is taking place. When a host sends that message, it includes its own MAC=IP information. This process is called **Address Resolution Protocol(ARP)**. It is a critical piece to IP networks being able to function. Every host that is using IP on the network must have a table that keeps a list of MAC=IP addresses. In other words, an **ARP table**. Memorize that, you will use it when it comes to troubleshooting network problems often. The other important table to remember is the **MAC table**.

ARP table = **MAC to IP** - This is how to identify which computer has which IP address.
MAC table = **MAC to port** - This is how to identify which MAC address is seen on which port.

I cannot stress enough how important it is to be comfortable with these concepts.

Sidetrack:
We had a great deal of difficulty one day.

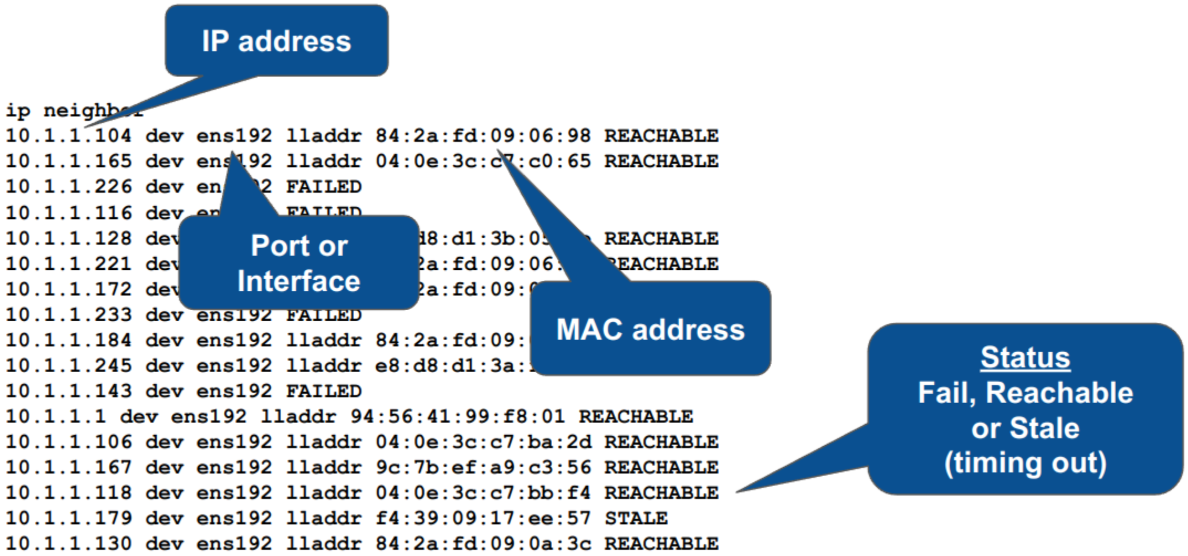
The help desk started to get calls about computers suddenly not working on the network. It was happening one by one. “It was the strangest thing”. This was a network for a large building and had a couple hundred computers in lots of offices. We rushed over there.

When I got to the computer, I started with the basics. Can I ping? Can I ping the gateway? When neither worked, I checked to see if it had an IP address. It did, but it was not in the network it was supposed to be. When computers start up, they usually automatically get an IP address through a process called DHCP. That takes a DHCP server that will answer a request for an IP sent to FF:FF:FF:FF:FF:FF. This was a Windows computer, so I could type “ipconfig /all” at the command prompt and along with the MAC address and IP address, it gave me the IP address of the DHCP server. Now, let’s look at how this all comes together.

1. At the Windows command prompt, I entered “arp -a” for a table of IP addresses to MAC addresses. In other words, I checked the **ARP table** of the workstation. If an address is not there and you know it is on the local network, try to ping it and then check the table again. If it is there, any IP attempt to connect will issue an ARP.
2. With the MAC address of the DHCP server, I could then check the switch and look at its MAC table. Because we took care to enter the description of each room and port combination for each port on the switch, I knew right away which room the DHCP server was in. Someone had brought in something that was acting as a DHCP server, and giving out responses to other workstations faster than our DHCP server. That is why the workstations were coming up with different addresses, addresses that did not route on our network.
3. I was then able to go to that room and sure enough, a contractor had wanted more ethernet ports in that office, so they brought in a Linksys router from home. The problem was, it was seeing the broadcasts of everything on the network and happily handing out IP addresses to every host that asked.

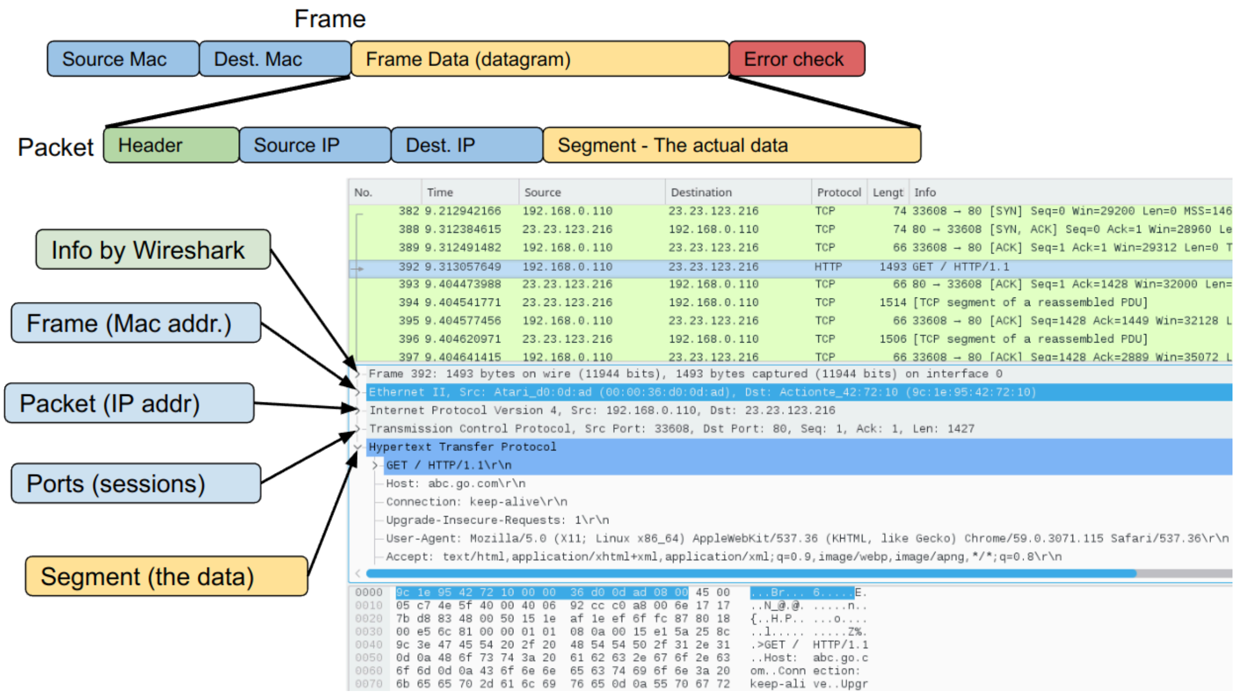
Knowing the basics are invaluable to troubleshoot. When you have the IP, ARP will give you the MAC. When you have the MAC, MAC tables can give you the port. Every host that does IP will have both. They are two different tables even though a utility might give you all the information at once.

Some commands will combine the information from both tables.



Part 2

Technologies in play



Now that we are getting a little more comfortable with the idea of how Layer 2 MAC addresses interact with Layer 3 IP addresses, let's take a more of a look at how this can play out with a few common solutions.

If you are getting into technology and your curiosity has led you down the very healthy path of exploring computers, chances are pretty good that some of you have installed an operating system. Perhaps you are the person family and friends call for tech support, maybe you have even done some work fixing other's computers. If you have installed an operating system in the past it was probably with either a DVD or a USB drive. It becomes a matter of booting off of the media, formatting the disk partition, and then installing the OS. Okay, that is definitely a good way to do it, …until you have to do it for tens or hundreds of computers at a time. Then you start to enter the world of **Enterprise Computing**. Simply put, an Enterprise solution is when you use technology on a larger scale that can meet the needs of a large organization. The example here that is a good illustration of how important networking concepts can be is making use of **Desktop Management**. If you have installed an OS you will quickly shudder at the prospect of going around machine to machine with a pocket full of USB drives. So, this is another way to approach the problem, utilizing the Preboot **Execution Environment (PXE)** and **Wake On LAN (WOL)**. Before that however, we need to talk about **Dynamic Host Configuration Protocol (DHCP)**.

DHCP

When you boot up your computer at home have you ever thought about how it gets an IP address? You might have heard the term, or have even used the term DHCP, but have you really thought about how it works? It grew, like most protocols, directly as a need to solve a problem. When IP was getting established, addresses were basically assigned. A network administrator would keep a list of the hosts, and when asked would give the operators an IP address for that host to use. That works when there are only a handful of hosts on the network. In response to that problem some folks at Stanford came up with the bootp protocol. That was a protocol that would read the assigned database that the network person filled out, to give an IP address to a workstation based upon its MAC address. The IP address would not have to be configured on the OS, but whoever had the computer, had to give the network person their MAC address.

Well, you can quickly see how that became hard to manage. You could not get an address until your computer was entered into the database. To prevent running out of addresses, the network person would have to do an audit to see if the IP addresses were still in use. Only then could they free up that address and give it to someone else. As computers became more popular, that broke down real fast.

Enter DHCP. Some folks at Bucknell University proposed this solution. Instead of simply giving out an address until it was revoked, give out addresses with a lease. You can easily see this in Windows with an “ipconfig /all” command. Half way through the lease time, the host will ask the DHCP server to renew the lease for another lease period. If that does not happen during the lease time, the DHCP server will then give the IP to another host upon request. The DHCP server can give a lot more information than just an IP. It can also give information in the form of options. These options can list things like the default gateway, the DNS server, and much, much more. One of the valuable bits of information can include the PXE server. For an idea of how many can be used, you can visit the IANA site for the standard options. There can be many more custom ones that are not listed.
https://www.iana.org/assignments/bootp-dhcp-parameters/bootp-dhcp-parameters.xhtml

But how does it work? Remember how broadcasts work at Layer 2? Whenever a host sends a DHCP request asking for an IP address, it sends it to FF:FF:FF:FF:FF:FF. The DHCP server or router will pick up that broadcast taking note of the source MAC address. It then checks its database. It keeps a list of what IPs it has given to which source MAC address, and the lease time. There are times when an administrator will want a specific computer to have a specific IP address. That can be set up in the database. Even if the lease has run out, if the DHCP server has not used up all of its addresses, it often keeps the past MAC=IP combination in the database and will give it the same IP again. Once the server has given an IP address to a host, then communication happens over IP instead of just using MAC addresses.

Desktop Management

To install an operating system, you have to first be able to load a small version of an OS to control the computer. After all, the primary duty of an operating system is what the kernel

provides. That is the portion that is loaded into memory on a computer and controls all the basic elements of a computer. It controls the CPU and flow of information in and out of memory. It provides the operations of all devices at the base level. It is the basic part that allows other programs to run and use the hardware. When you are installing, that is all that is really needed. Installers that come with a USB will have more to provide a nice interface in which to ask you questions and get information, but to install, you don't really need all that. You simply have to load the necessary elements into memory. A network boot allows you to load the basic OS from a server instead of a disk drive. Once that is in memory it can control the disk and communicate over the network. That allows for a machine to format the drive, and copy whatever files it needs to the computer from the network.

Steps to install from a PXE environment providing desktop management:

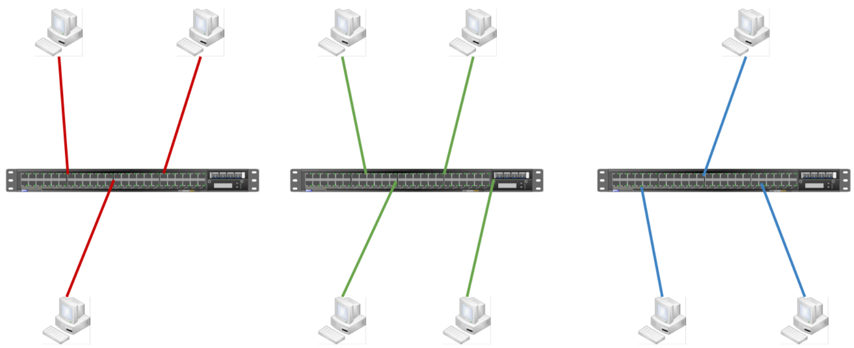
- Set the BIOS to network boot
- The network card makes a DHCP request
- The DHCP server provides an IP address to the host and gives a PXE server address in the options
- The BIOS loads the OS into memory from the server
- The OS loaded into memory has what it needs to get instructions, if the instructions are to image the machine, then it formats the host hard drive and starts a file copy of the new image.
- Once the new image is copied, it reboots.

The big advantage of this process is that it does not have to be attended. You can set up a large number of computers to all be restored overnight, automatically. The part you have to be careful of, is that if your image has problems, you can also goof up a large number of computers overnight. Enterprise Desktop Management requires a great deal of care, testing, and learning fine details of operating systems and applications to be able to manage systems flawlessly. The main difficulty with overnight implementations is that computers tend to be turned off. That is where Wake On LAN comes in.

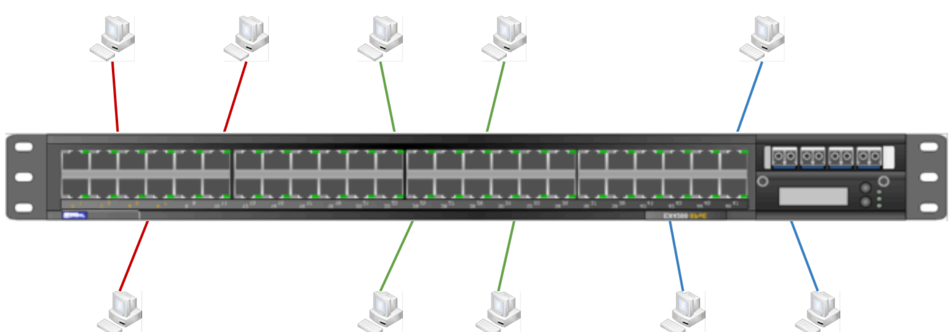
Modern network cards, when used for network booting, do not power down when the computer is turned off. They stay on. However, when the computers are turned off they do not hold on to an IP address and they stop communicating. They go into a listen only mode. They listen for a very specific type of frame coming across the wire. The frame has a destination MAC address of FF:FF:FF:FF:FF:FF, a Layer 2 broadcast. In addition, in its payload it also has 16 repetitions of the network card's MAC address. When a network card sees that, it basically recognizes "Oh, that's me!" and turns the computer on and goes through the network boot process. That is another example of again, how important understanding the relationships between Layer 2 and Layer 3 is. Remember reading about the Layer 2 broadcast domain? When a frame is sent to the broadcast MAC address everyone on that network sees it. Notice how many processes required the ability to send to the broadcast. Having a good understanding of the basics are invaluable in troubleshooting and having an awareness of how a hacker can manipulate the network.

VLANS

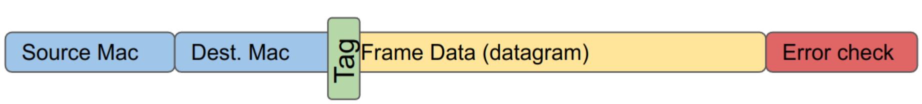
A lot can happen on a local network. IP addresses are assigned, ARP controls which MAC address is seen as being the owner of which IP, and computers can be started when they are turned off. If you are on the same broadcast domain, even if a host is firewalled to not answer any IP packet sent to it, if it still wants to communicate over IP it will answer an ARP. You only see the ARP requests and answers on the broadcast domain. With all that going on, one of the basic principles for network security is **Network Segmentation**. It simply makes sense to completely separate the networks.



The first and most obvious way to do that is to simply buy more switches. If a workstation is not on the same switch, and they are not interconnected, they will not allow any communication between networks unless there is a router in between. However, as switches can be very expensive, the concept of vlans were developed. VLANs or **Virtual Local Area Networks**, were developed by adding a tag to the frame. That tag basically designates which network that frame belongs to. Then, with the same switch, you can assign ports to specific VLANs to achieve network segmentation of broadcast domains.

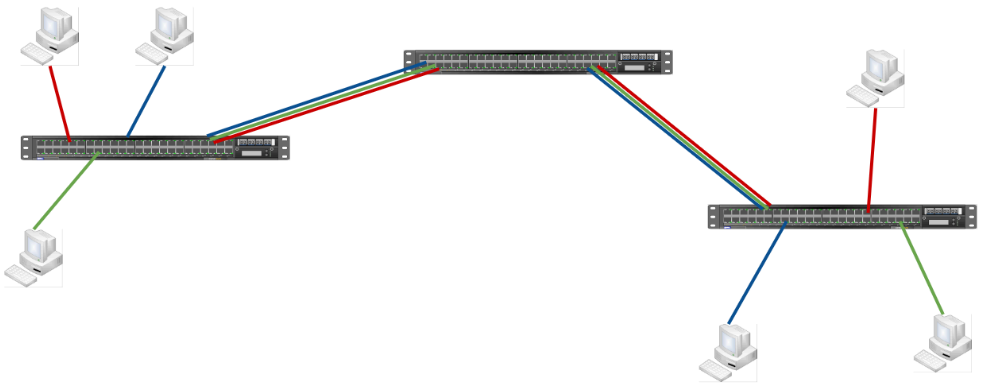


When the frame enters the switch it gets a tag based upon which VLAN that port was assigned. The tags are actually numbers, but in this case to make the concept easier I am using colors.



If it came from a host on the green VLAN, then if that host sends a broadcast, only those that are on the green VLAN would see the broadcast. Once the frame leaves the switch the tag is removed and the host has no indication that it was ever segmented from the red and blue

networks. Hosts and equipment that are assigned an “access” port only see the traffic for that specific VLAN. However there are times when a port needs to connect switches or provide a path for multiple VLANs. That is known as a “trunk” or “tagged” port and is commonly used to connect network devices or virtual hosts. When a frame travels across a trunk port it maintains the tags on each frame so that the switch on the other end can tell which VLAN the frame belongs to.

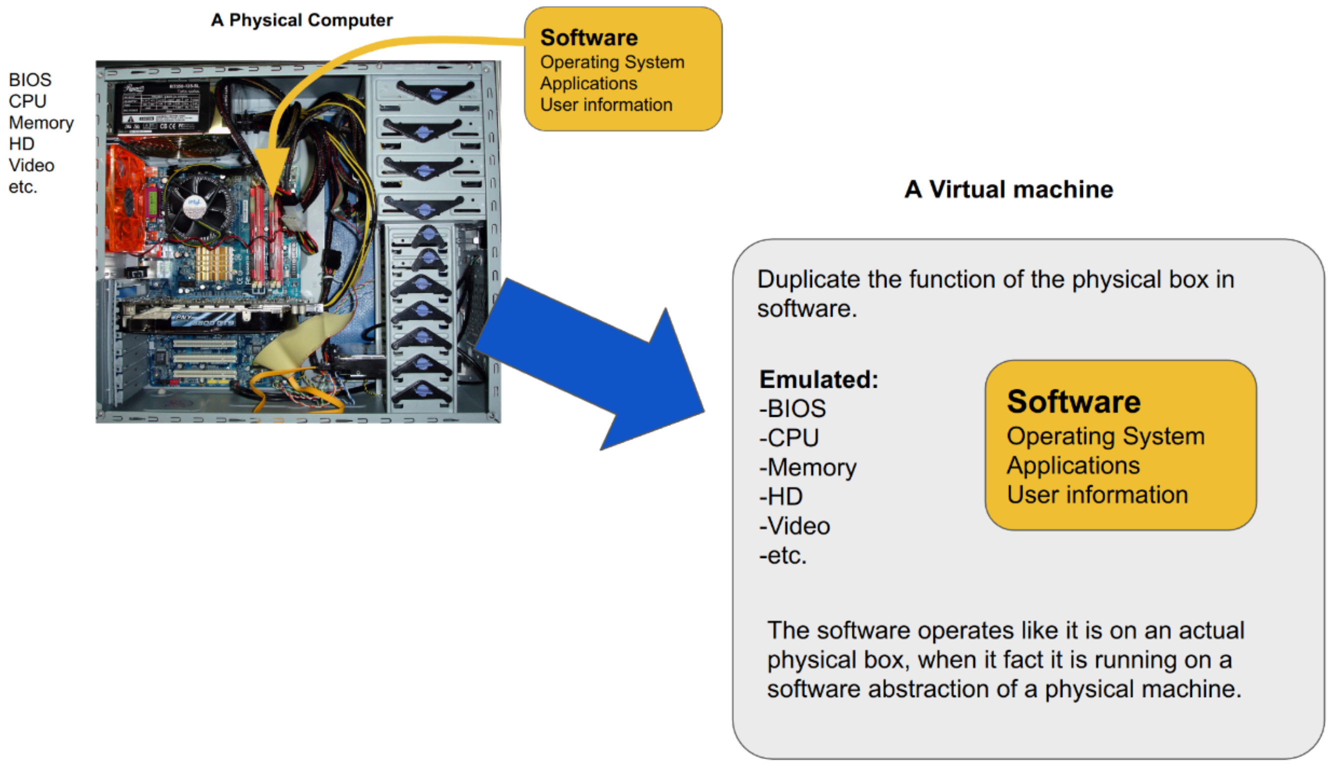


VLANs are Layer 2 based. Remember, their function is to keep broadcast domains separate. When you send a MAC address of FF:FF:FF:FF:FF:FF (broadcast) everyone on that VLAN will see it. One of the more important areas where this is used is in the concept of virtual machines. I am not going to go too much into detail about virtual machines, but let’s give a bit of a reminder.

Virtual Machines

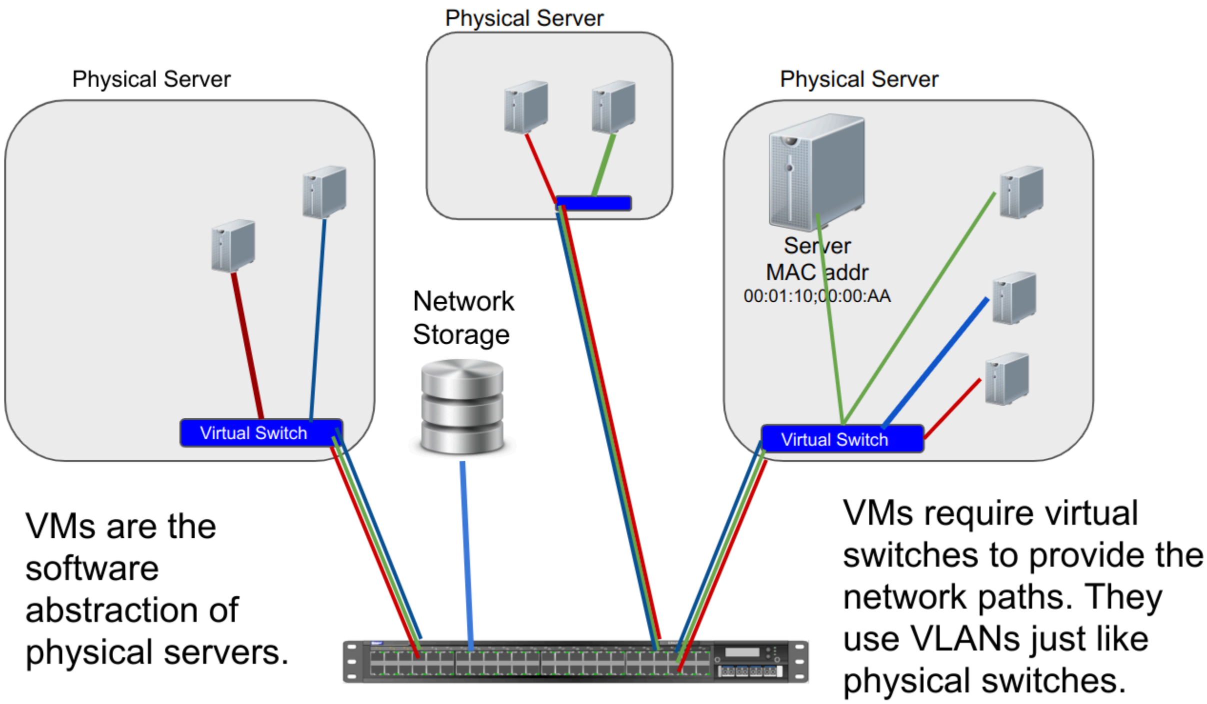
Virtual machines are cool. They have had a major transformation on technology and are invaluable to students who are learning how all this stuff works. If you have not explored VMs, I highly encourage you to do so. There are some fantastic free tools out there. Virtualbox is a free hypervisor which allows you to host Virtual Machines (VMs). If you are running Linux there are free options as well in KVM and Xen. Modern hypervisors are not limited to virtualizing just computers, but networks as well.

A quick reminder:
A Virtual Machine is basically taking everything that a computer is and the way it behaves, and duplicating it in software. Once a VM is created, you can adjust all of its attributes as long as you don’t exceed the capabilities of the physical machine it is running on. From earlier classes, when you installed an operating system and programs on a computer, you had to install everything onto a hard drive. You might have had to change settings in the BIOS, make sure you had a capable CPU, had enough memory, etc. A VM takes all of those physical aspects and duplicates them in software.



The big advantage of a VM is when the physical box that is running all the VMs, instead of having a whole bunch of idle processes where the CPU is just sitting around with nothing to do at the moment, can now spread those resources across multiple VMs. This saves tremendous amounts of energy, physical hardware, and reduces heat. For large data centers that can mean the difference of millions of dollars over a short period of time. Whenever you do not have to duplicate physical boxes, you save money. VMs can also provide stability.

VMs and layers of abstraction



Technology seems to be getting more and more complex. Once you get comfortable with one topic like what a computer is, or what a network is, it then gets abstracted into software. Take this image for example. Here we have physical servers represented by large boxes, with virtual machines running inside them. Not only that, but each virtual machine has its own operating system and programs actively running inside. In real life what that means is, the memory in a virtual machine is in fact just a portion of the memory on the physical server. The CPU core(s) a virtual machine is running is actually a portion of the cores that the physical server is running. Even the storage is virtualized. The operating system that is running on the virtual machine thinks it has its own disk. In fact however, the “virtual disk” is actually a small portion of the overall storage and is not actually on the physical server. In this drawing, it is actually a separate disk store and accessed through the network. The HUGE advantage to this is, you can move a virtual machine from one physical server to another. You don’t have to copy the data. All you have to do is redirect the flow of the data. This can even be done while the virtual machine is running. A web server running on a VM can be transferred from one physical machine to another without having any idea that anything happened. If one physical machine starts to get too busy because some of the VMs are using too many resources, the other VMs can be moved in real time to a less busy server. When you put all of that together it can be a bit to wrap your mind around.

Virtualization and networking

To make this work, physical hosts are housing multiple VMs. That increases the networking complexities. VLANs were created to segment networks. This is a good idea for many reasons

with security being one of the most important. Since VMs represent separate computers, part of their virtual hardware are network connections. If we have multiple computers, on multiple VLANs, then on the physical servers we will also need to have “virtual” switches. These are the layers of abstraction in operation.

Sidetrack:
When you approach technology, you will find it invaluable to take the time to learn the basic concepts. As you go through your IT career, never be satisfied simply learning what setting to change. Always ask yourself, what exactly is happening here, and take the time to make sure you understand the concepts. The primary reason is, often these concepts will simply become more complex and stack upon each other. If you take the time to get them down as you go, your journey will be much easier. It will make sense. It is easier to remember information when you have the conceptual understanding to hang that information upon. As you get mileage under your belt, you will start to see more and more relationships. Learning new tech becomes easier because you now start to know what to look for. You know that for something to work, it will have to be able to provide the same functionality as some of the other products you have worked on in the past. To be able to do that,...you need to know the concepts.

You will also realize that many of those individuals that you learned from, and thought were incredibly brilliant, were not so different from you. They just had more mileage.

Now let’s bring some of that together. We have learned how important broadcasts on Layer 2 can be.

- Whenever a workstation needs to contact another host on the same IP network it needs to send an ARP to build its ARP table so it knows which ARP=IP. It does that by a broadcast on the broadcast domain.
- Now for a VM in the virtual sphere, the switch needs to contain those broadcasts based upon which VLAN it is in. That is done by the virtual switch.
- The virtual switch has to pass on the frame WITH the VLAN tag to connected physical switches so they can keep the information intact.
- When the virtual switches connect to the physical switches, and the physical switches need to connect to the next physical host that houses a virtual switch, they use trunk ports to keep VLAN tags intact.
- The virtual switches need to keep a MAC table (MAC=Which port) just like the physical switches so they know how to pass the information to which VM inside the hypervisor environment.
- When VMs move from one physical server to another, the MAC tables of where that MAC address is, needs to be updated in the old virtual switch, the physical switch, and the new virtual switch almost instantly.

Whew! Just think, that is all just so Layer 2 can get a frame in the right place. That is all still on the same network and VLAN.

A reminder:

- Everything passes in frames. It must have a source and destination MAC address.
- The method of matching MAC to IP addresses is through ARP.
- ARP tables match IP to MAC in a broadcast domain. To go from one IP network to another, requires routing.
- VLANs form separate broadcast domains. If you want to go from one VLAN to another, you will have to use a routing process.

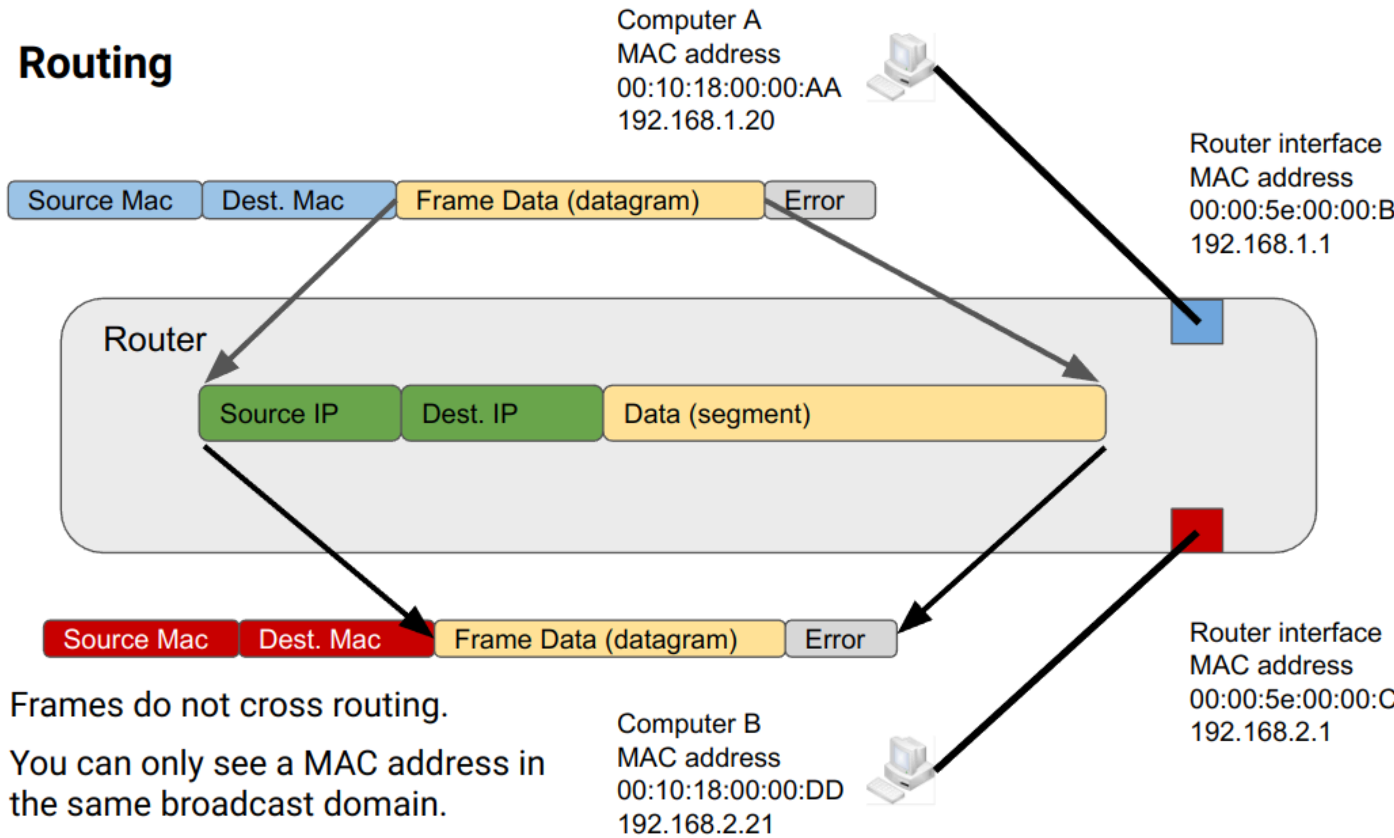
IP Routing

We have covered how to get from one workstation to another in a network. Now let's look at the reasoning behind IP. Remember from its origins, IP was created to get beyond the local network and provide a scalable method of communications between networks across great distances. The MAC address and its broadcasts work on a local network, but there needed to be something that would persist across different types of networks and infrastructures. The solution was to encapsulate this packet into a frame.

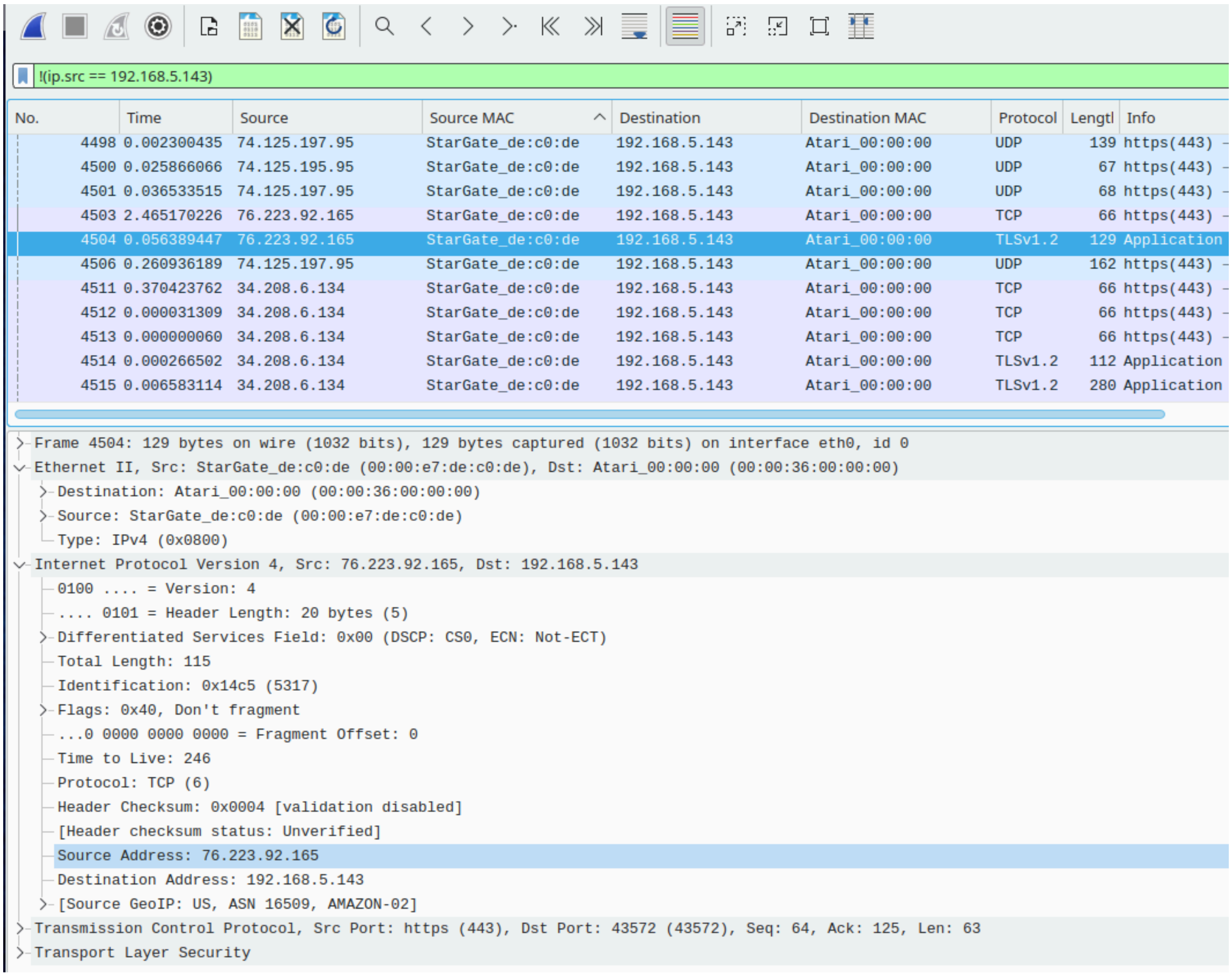
Think of it this way. The Post Office makes a great analogy. When you post a letter, you put it in an envelope and give it a To: and From: address.

1. The mail carrier picks up your letter and puts it in a bin. The bin represents the frame.
2. The bin is emptied and letters are sorted. That represents routing.
3. Letters are put into another bin and sent to a post office. - Another frame.
4. It gets dumped at that Post office and sorted. - routing.
5. It gets put into a different bin to be placed on a mail carrier’s truck.
6. It is put into the mailbox of the person you sent the letter to.

The bins change along the way (frames), but the letter’s addresses and envelope (packet) do not.



In the diagram, Frames are represented by color to emphasize that once you go through a routing process, the MAC addresses are stripped off to then be inserted into a new frame on the next network. A router sits with a port in each broadcast domain. Everything that wants to talk to a different network will be sending and receiving all packets from the same MAC address, the MAC address of the router. If you pull up Wireshark and look at network traffic you can see this:



In this screen capture, you will notice that I filtered everything by removing packets that had my IP as a source address. I also included columns that display the source and destination MAC addresses. I spoofed my MAC addresses to sanitize publishing this information. You will notice that my workstation has a MAC address that starts with 00:00:36 which is the OUI for Atari. My router I spoofed with 00:00:e7 which is the Stargate corporation. Notice there are a few different IP addresses that have the same MAC. All that means is I can't see their real MAC address because MAC addresses change as they route from network to network because frames do not persist. Packets persist, and that is why we needed a different type of address when going from network to network.

The routing basics

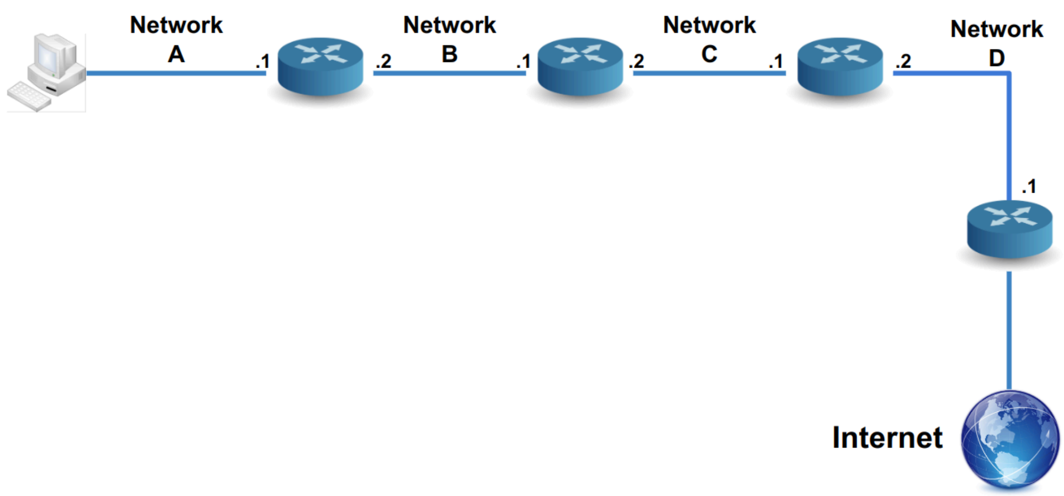
If you have ever manually configured an IP address on a computer, you will probably already be familiar with the three things required for IP to work well.

IP addresses: Every computer that is going to communicate with the Internet Protocol needs a unique address. That can be a private address behind a device doing Network Address Translation (NAT), but we will get to that later.

Subnet Mask: A subnet mask only does one thing. It designates which part of the IP address represents the network and which part represents hosts on that network. Do not read anything else into that. That is all it does.

Default Gateway: The basic concept of a default gateway is, if a host does not know the proper path to take, then send it to a device that should know the path.

Routing - Getting from Host to Host



Now let's try to simplify routing. The basic concept is that a router (or routing process) will take packets from one network and send it to another. Routers are the connectors of networks, and as such, have an IP address in each network they are connected to. So if a packet is sent to them, they have to make a choice as to where to send the packet. If the destination is a host on one of their networks, they simply send an ARP, find the MAC address of that host and send it there. If the destination is not on one of the router's networks, then it needs to send it on to the next router.

To make the explanation easier, let's not use full IP addresses. Instead let's just call the networks by letters with host numbers. In the diagram, we have a computer on the "A" network, connected to the router via A.1, the other side of the router is B.2 which connects to the interface of the next router B.1. We keep following that pattern through network D to get to the Internet.

When the host wants to get to www.somewhere.com it will go to its Domain Name System (DNS) server and translate www.somewhere.com into an IP address. We will delve into DNS later. Once it has an IP address for a destination it first looks at its subnet mask to check "Is that

on my network?”. Then when it is not on its network, it sends the packet to the default gateway. In other words, think of it as the workstation thinking “I don’t know where to send it, so I will send it to someone who is supposed to know”.

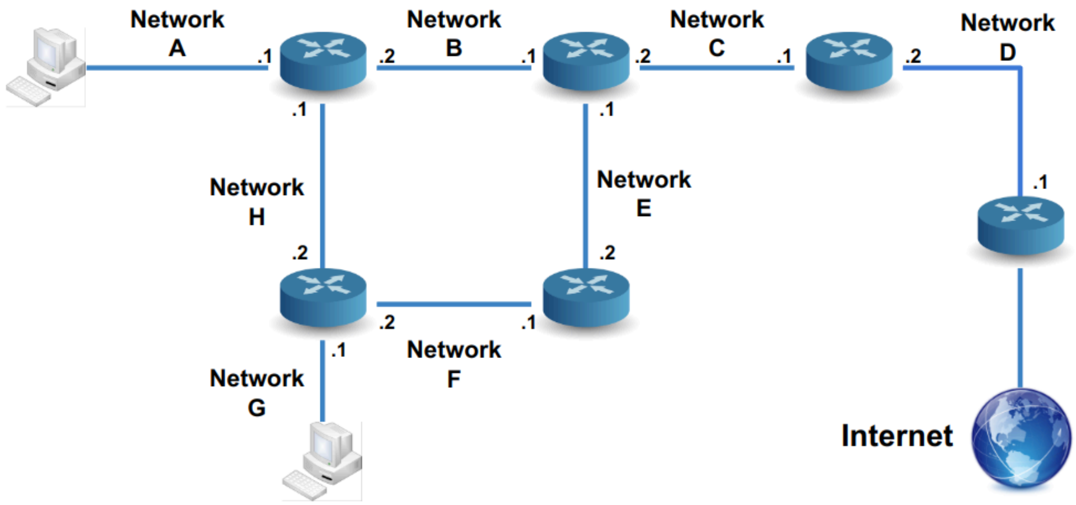
The default gateway (router) gets the packet, checks the networks it is attached to and their subnet masks to determine if it is connected. Remember that if it is connected, then it will send an ARP to find the MAC address. If it is not connected then routers also have a default gateway. That is almost always the path that leads to the Internet. So in this case the router’s interface on the B network is B.2. It sends the packet to B.1 which is its “default gateway”. The second router gets the packet, goes through the same process and upon having no idea where that network is, sends the packet to its default gateway which is C.1. See the pattern? The routers keep sending to default gateways till it gets to an Internet router. Internet routers have very large routing tables. Their routing tables have millions of entries that definitely give the route to take for the packet to get to its destination. It is the way back where things start to get interesting.

The last interface we hit was our “border router”. That is basically the demarcation point, or the point where the infrastructure is no longer our company, but that of the Internet Service Provider (ISP). The packet went through networks A, B, C, and D to get out. When it comes back and the host on network A is now the destination, then each router needs to know where to send the packet. The first router from the Internet, with the interface D.1, must have a routing entry or it will send it to,..the default route. Remember, if a device does not know, that is what the default route is for. So the packet comes in, with now instruction the router will send it right back out, to which that router will send it right back in. Well, as you can expect the return packet kind of dies right there. The solution is the first router MUST have a table that tells it, to get to network A, send it inside. For that router that would be sending it to D.2.

When the router with D.2 gets the packet, it checks its networks of C and D, not seeing A, sends it right back out to its …default route. So, for the return packet to get where it needs to go, the router must have a routing table entry telling it to get to network A send it to C.2. See the pattern? The basic routing concept is fairly simple. If where the packet is trying to go is not on the default route, then there needs to be an entry pointing to the next step to that destination. For it to work, then all the routers along the path need to also have those entries describing the next step toward the destination. They all go through the same decision process.

- Is the destination connected to the network? (check interface IP addresses and subnet masks) if yes then send an ARP to find the MAC address, if no then check the routing table for the next hop.
- If unknown, then send it to the default gateway.
- For the next hop (which has to be off of one of the router’s interfaces), send an ARP to get the MAC address of the next router.

A simple path is easy. When you get multiple paths to take to get to a destination is when it gets interesting.



When there are multiple paths to take in a network, that is where routing protocols come into play. Routing protocols are basically the communication between routers so they can compare notes of which network is connected to whom. That way if there is a break in the network, and traffic needs to take an alternate route, the routers can alert each other to detour the traffic. The complexity builds when you throw in other factors as well. A shorter number of hops might not be the best path to take. A slightly longer path might be a lot better if it is going over faster links. Even with more complexity however, routers are still doing the same task of sending a packet onto the next hop, all based upon their own table which has a very limited perspective.

Part 3

The State of Things

So far we have looked at the basics of how to get a packet from point A to point B, now let's take a look at how to keep track of the data. Again, I would like to keep things in perspective by looking at the history. When you want to send data from point A to point B, one side has to be running some type of program that will accept the data, or open a path for a conversation. When you look at this on a programming level, from early days, it is known as a “socket”. It is named a socket inspired by the concept of a physical cable connection. Once programs on both sides of the conversation connect, they don’t really care about the network. The program just wants to send data to and from the established connection (socket). On an IP network that is where the concept of ports comes into play.

Sidetrack:
Please remember the context where all this was happening. The “Internet” as it was in the 70s was just an interconnection of institutions, mainly higher ed, that was trying to develop a more manageable connectivity of computers. Lots of folks saw the need, and things were evolving fast. While the international bodies were working on developing the standard that we were all to follow (The OSI), the US institutions were throwing ideas back and forth with constant updates of “Hey everybody, this item is getting out of hand so let’s all start doing it this way as a fix” approach. (Request for Comments-RFCs) As those updates kept passing back and forth, a protocol evolved. That is why we are all using the Internet Protocol (IP) instead of the OSI protocol, which is what the international teams wanted. A good example of one of these updates was the agreement of ports. Do you remember from previous classes having to memorize some of the “standard ports”? That is all because Jon Postel and folks asked for a catalog of what everybody was using at the time and issued an RFC basically saying let’s start doing it this way in [RFC 322](https://www.rfc-editor.org/refs/ref0322.txt).[6] [7] [8] However, addresses and ports were very different. These ports were the “AEN” number in an earlier sidetrack. Remember the “Another Eight bit Number”? Why is this important? I want you to realize that IT and computing is in a constant state of flux and not as well defined and engineered as some of your books might lead you to believe. **YOU can make a difference!**

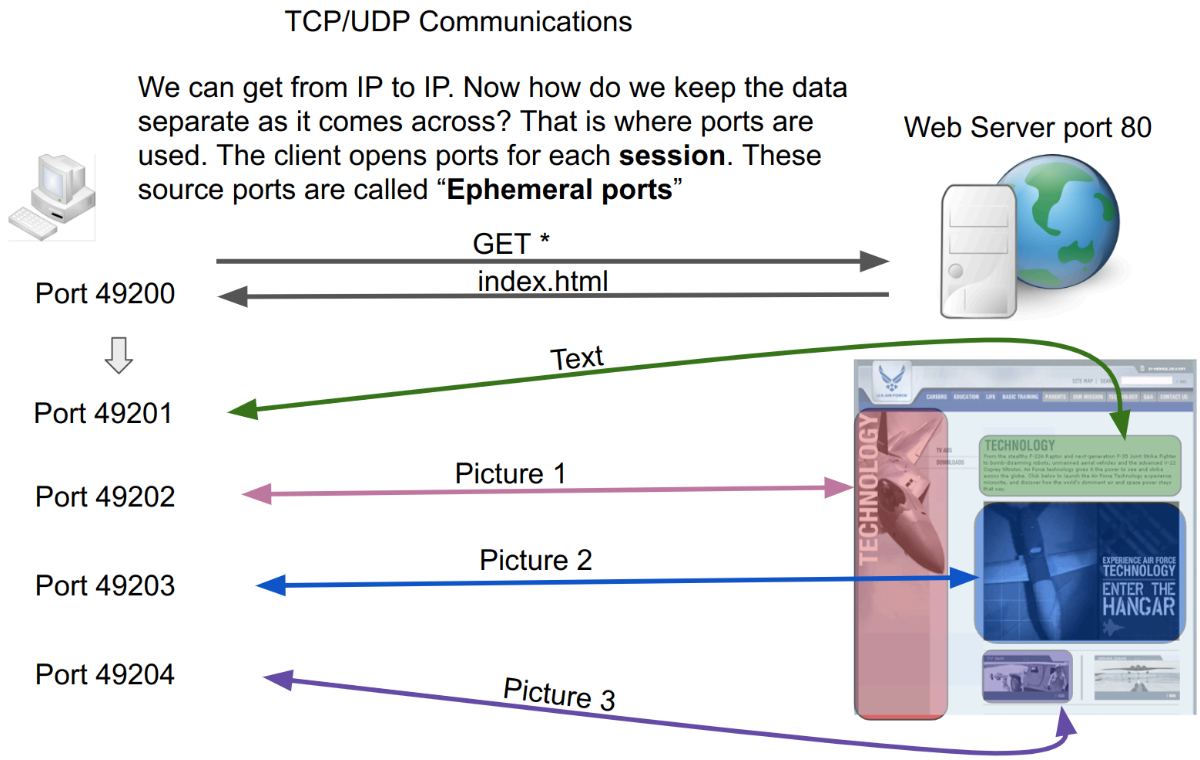
IP addresses are what gets packets from one point to another, but for communication to work, an application/server needs to be running to receive the information sent to it. You can have multiple applications running on the same IP, so to keep conversations separate, ports were developed. An application, like a web server for instance, is listening on a port so that when

[6] "TXT - » RFC Editor." https://www.rfc-editor.org/refs/ref0322.txt. Accessed 29 Jan. 2023.
[7] "RFC 349: Proposed Standard Socket Numbers." https://www.rfc-editor.org/rfc/rfc349. Accessed 29 Jan. 2023.
[8] "RFC 433: Socket number list." https://www.rfc-editor.org/rfc/rfc433. Accessed 29 Jan. 2023.

data comes its way it will process that information correctly and in turn respond correctly. When the server application starts, it takes a port to listen on. You can have an application run on any port you want. There is no rule or enforcement that controls ports. An application just can't use a port that is already in use.

Standard ports and ephemeral ports, let's have that conversation:
You might be thinking right now, "what about all those ports we learned about in previous classes?" We learned that web servers are on port 80. We learned that secure web services like Transport Layer Security (TLS) is https: and on port 443. That is all true, but those are standard ports. What that really means is those are simply ports that everyone agrees should be used.

The advantage of standard ports is that you can have an application like a web browser go to www.somewhere.com and you don't have to add a port to every web page. If it was on a different port then you would have to type in something like www.somewhere.com:8080. Having standard ports makes it a lot easier. From a hackers perspective however, that can be completely irrelevant. Just because something is allowed to port 80, it does not mean that it has to be a web server. If you have ever set a firewall rule, unless you are using an application based firewall, you probably set a rule to allow any address to go to a specific IP address on port 80. That's great,...unless you were trying to only allow web traffic, and instead, it was a completely different application like a Remote Access Trojan (RAT). If you want to get access through a basic firewall, the best way to do that is to put your malware on a standard port which is allowed to pass. When we start to look at network communication, the standard destination port is often not as important as the conversations, and where they are coming from. When we start to look at Wireshark, it can be very confusing. There is a LOT of data intermixed. The only thing that separates the conversations is more importantly the source port, also known as the **ephemeral** port. Let me give an example:

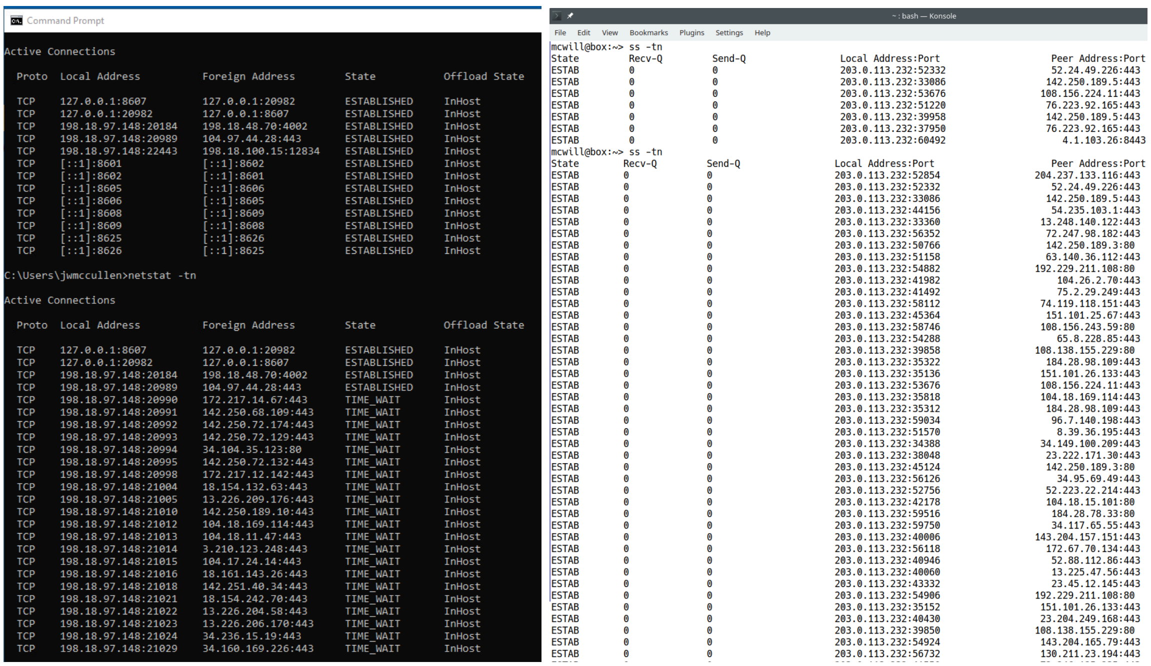


When you go to a web server, your browser will initiate a connection. It will use TCP which has a three way handshake (which we will go over later) that will start the connection and then ask for the web page. The server then responds with the index page that basically gives the initial layout of the web page. I can include information like how many pictures, text and where to download them from. It will give the overall layout of where those pieces of information will fit. With that information, the workstation starts multiple sessions/conversations asking for all those elements separately. They might all go to port 80 on the server, but that is just a small portion of what is going on. As the data streams in as partial chunks of each item they will not be all together. It is which ephemeral port the server responds to that allows the client to keep the data for each item separate. On top of that, often the items on the web page that the index page lays out for your browser, might not even be from the same server. You can go to cnn.com for example. You might trust cnn.com, but some of that content is likely not even coming from cnn, but cnn’s advertisers. Do you trust them? There is a term for malware that is delivered through compromised advertising servers. It is malvertising. When you are looking at wireshark, all this data comes streaming in at once and requires the use of display filters to start to make sense of it all. It can be quite an eye opener.

Sidetrack:

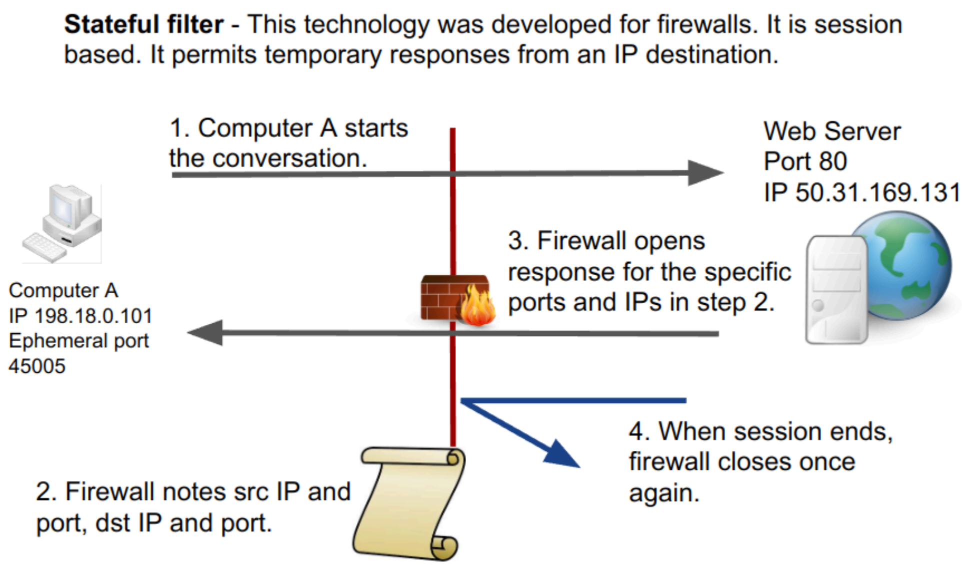
If you are on windows, you can open a command prompt window or a terminal in Linux. To see your current sessions enter the command “netstat -tn” in Windows, or “ss -tn” in Linux. That will show all your current tcp connections. Now try opening a browser and go to cnn.com. Run the commands a second time. With all the content that you get from a site like cnn.com, you might

be quite surprised to notice just how many separate sessions/conversations/streams are involved. Each one represents a momentary socket from your computer to the web server. Also, please notice how many different IPs you are visiting. Remember, it is the combination of your IP and ephemeral port that distinguishes each separate conversation you are having from the web server. By the way, conversations, sessions, and streams are all synonymous. Display filters in Wireshark use the term stream. Many firewalls will count their capacity in session counts. Just know that when you get to looking at network information and threats, this is one of the most important concepts to understand, because so many different technologies rely on it.



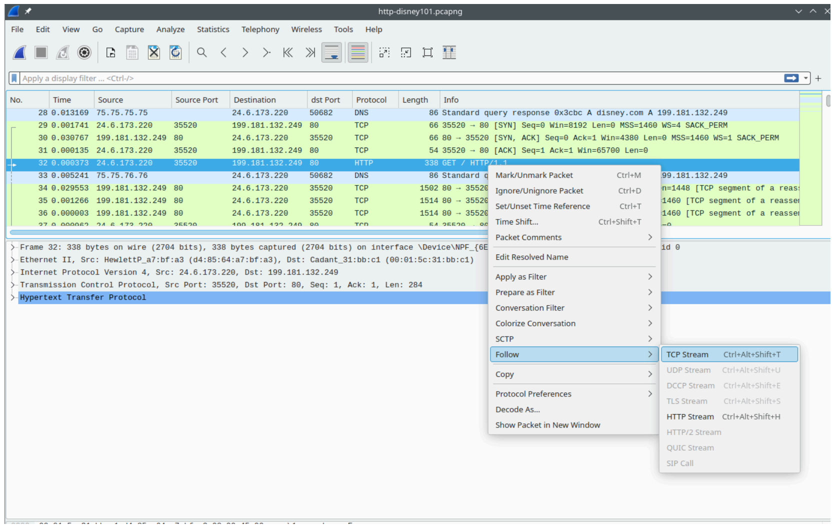
The Stateful Firewall

There was a huge advancement in 1989 made by ATT labs when they developed the session based, or stateful firewall. Before that firewalls simply were access control lists. These were highly insecure as they permitted specific standard ports out, but had to allow all ephemeral ports access in. So to bypass the firewall, all you had to do is put a server of any type on any ephemeral port. Keep in mind at this time compute power was minimal. This made it very difficult to put more intense processing on routers and firewalls. By 1994 Checkpoint came out with the first stateful firewall that really started to make a difference in the market. This is how stateful firewalling works:



It is all based upon the session. A computer starts a conversation out to the Internet. As we have covered, the way that works is for the source to open an ephemeral port. That provides the firewall with both the source IP and port. The assumption is that there will certainly be a response from the destination IP and port back to the source. The firewall keeps track of this conversation. After the first packet goes out, the firewall then opens a temporary rule for the return IP and port of the server to the specific IP and ephemeral port of the sender. Once the conversation finishes, which is usually by a FIN (finish), RST (reset), or time out, the firewall once again blocks all incoming traffic.

The important thing to grasp is that all of this happens based upon the individual conversations. Sessions live and die very quickly. When you visit a web page, once all the items are downloaded, that represents the life requirement of the sessions. Other sessions are longer. Downloading files for instance. Have you ever experienced a long download? That is an example of a longer session. In Wireshark we will cover display filters. There is a display filter that will allow us to view only the packets that represent a single session so that we can separate that traffic from all the rest, all from a right click on a packet that is part of the stream:



The advantage of Wireshark is that it has disectors that can interpret the data in packets to analyze what is being transferred across the wire. In this example it has figured out that the traffic is hypertext transfer protocol and when you follow the stream it will display the complete conversation. It is one thing for a firewall to block ports and IPs, but when it can watch the packets throughout the stream/session as they come through the firewall and identify an application based upon the data in the segment, then we are taking firewalling to a different level. These types of firewalls were given the name “Next Generation Firewalls” by the Gartner consulting firm in 2009. Next gen firewalls do application analysis and allow you to make firewall rules based upon the application instead of simply identifying ports. Keep in mind, this is all reliant upon the concept of streams/conversations. That is why this concept is so important.

Application Usage

APPLICATION	RISK	BYTES	SESSIONS	THREATS	CONTENT	URLS	USERS
ssl	4	1.1T	682.7k	58	0	0	2.3k
syslog	2	198.6G	40	0	0	0	1
ms-update	4	47.7G	36.4k	0	0	0	576
grammarly	3	45.6G	3.2k	0	0	0	32
windows-azure-base	1	39.8G	5.0k	0	0	0	50
web-browsing	4	34.9G	96.4k	777	0	0	1.5k
nfs	3	28.8G	294	0	0	0	1
dropbox-base	4	27.5G	325	0	0	0	11
google-base	4	19.4G	35.8k	0	0	0	547
xbox-live	3	14.8G	536	0	0	0	2
apple-update	3	11.6G	2.0k	0	0	0	31
paloalto-updates	2	9.4G	10.9k	0	0	0	3
steam	3	8.4G	58	0	0	0	1
quic	1	8.4G	246.1k	0	0	0	220
vmware	2	8.4G	42.5k	0	0	0	55
ldap	2	6.9G	492.5k	0	0	0	534
insufficient-data	1	5.9G	34.4M	0	0	0	137
apt-get	1	3.7G	498	0	0	0	24
dns-base	3	3.4G	8.6M	280	0	0	774
ipsec-esp-udp	2	3.0G	45	0	0	0	24
ms-ds-smbv3	3	2.7G	284.6k	0	0	0	528
rtcp	1	2.7G	21	0	0	0	9
wireguard	1	2.4G	27	0	0	0	4
vmware-view	2	2.0G	4.1k	0	0	0	482
dnf	2	1.4G	1.7k	0	0	0	3
ssh	4	1.2G	3.9k	0	0	0	14
icloud-base	2	1.2G	7.3k	0	0	0	32
apple-maps	1	1.1G	35.6k	0	0	0	35
kerberos	2	643.9M	199.3k	0	0	0	531
paloalto-dns-security	1	542.5M	938	0	0	0	3
gmail-base	4	418.7M	4.5k	0	0	0	48
google-update	3	344.8M	31	0	0	0	19
sharepoint-online	3	322.3M	2.3k	0	0	0	27
ms-local-security-management	1	317.2M	65.0k	0	0	0	527
twitter-base	3	237.1M	815	0	0	0	24
ms-office365-base	2	223.1M	10.5k	0	0	0	107
dns-over-https	3	213.5M	23.0k	0	0	0	40
signal-file-transfer	2	210.1M	294	0	0	0	11
active-directory-base	2	177.1M	31.5k	0	0	0	528
rsync	3	175.8M	35	0	0	0	1
msrpc-base	2	173.5M	100.1k	0	0	0	528
itunes-base	3	170.2M	4.9k	0	0	0	31
unknown-udp	1	167.7M	28.4k	0	0	0	71
paloalto-wildfire-cloud	2	161.7M	10.0k	0	0	0	2
google-chat	4	159.7M	280	0	0	0	35
pan-db-cloud	1	158.7M	1.3k	0	0	0	2
apple-push-notifications	1	158.0M	6.5k	0	0	0	29
dhcp	2	113.1M	155.4k	0	0	0	653
youtube-base	4	104.9M	816	0	0	0	75

This is an example of the applications running through the CoE firewall.

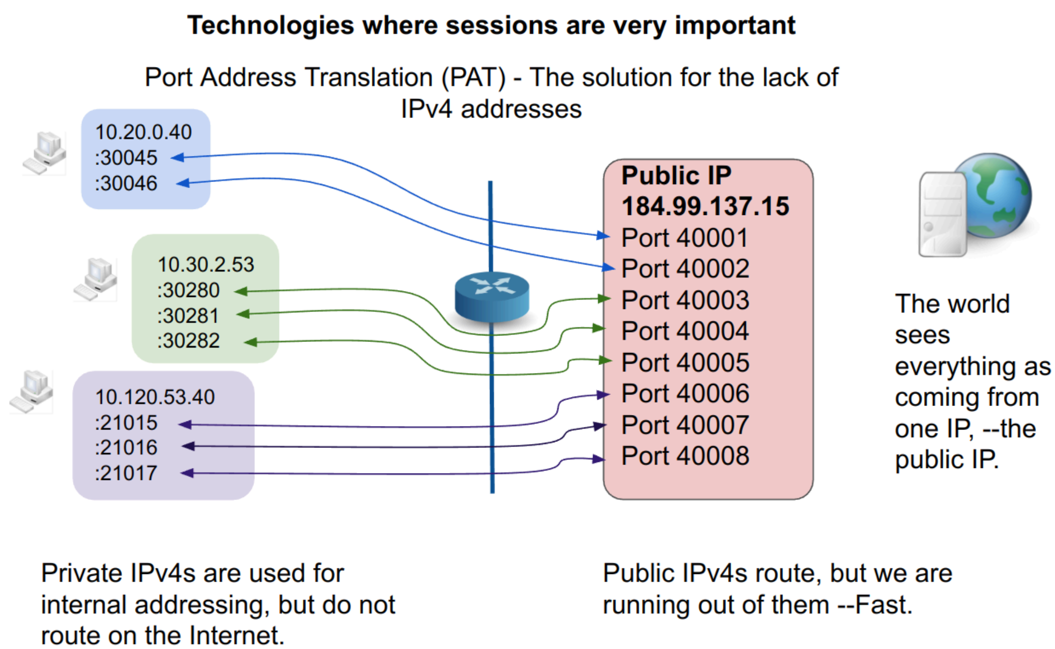
NAT and PAT

Another classic example of where sessions are heavily used is by a technology that we probably all use daily. That is Port Address Translation (PAT), which is a form of Network Address Translation. It also goes by the name of IP masquerading, many to one NAT, and a slew of other vendor names. I say a slew of other vendor names as there seems to be a problem in this industry when marketing gets too near the technology. You will find throughout this profession multiple names for the exact same concept, it just varies by product. Again, this is why it is important to learn the concepts and not just the names or what box to check. The focus of the moment however, is the importance of understanding sessions. They are used heavily in PAT.

Network Address Translation (NAT) was first designed as a method to handle some unique problems, but it rapidly became the solution for a much more common problem. The Internet was running out of IP addresses, and running out very quickly. As we will cover later on in this

document, originally all IP addresses came from the body that had oversight of the addressing space. That organization was the Internet Assigned Numbers Authority (IANA). Originally, organizations were handed blocks of IP addresses according to how many hosts they were going to have. When the World Wide Web took off, the demand for IPs exploded. This was an instant problem. IANA had to be protective of IP blocks. Organizations had no idea at their initial request for IPs as to how big their network would grow. There was an instant need for flexibility and reducing how many public IPs were in use.

Public IPs?
Yes, public IPs. Public IPs were IP addresses that could route across the Internet. The solution to the problem just described was to have Private IPs. Private IPs gave a huge amount of flexibility, they just had to be translated into public IPs. Let me explain. The solution to this problem was the creation of several groups of IP address pools that could be used by any organization. Anyone could use them, the caveat was that they could not route across the Internet. One of the first tenants of this whole networking thing across the Internet is that each host has to have a unique address, otherwise if addresses are not unique, how do you know who to send it to? The answer was in Network Address Translation. Once packets get to the edge of your network, that point where it heads out of your controlled space and out onto the internet, the private addresses need to be translated to public addresses. That translation also has to allow for many private addresses to use one (or just a few) public addresses. The reason this works is because of the concept of sessions.



Remember that the whole point of ephemeral ports is to keep conversations separate. So, all a device that performs PAT has to do is change the portion of the packet that holds a private IP and ephemeral port to the public IP and ephemeral port on the Internet side of the PAT device. For each session, it has to keep track of what private IP/port equals the public IP/port assigned in its table. The world just sees the Public IP and ports. When the packets come back to the Public IP/port, the PAT device changes the information in the packet to the private IP/port. The

reason this works is because every IP has 65,535 ports to work with. That is a lot of sessions. Remember sessions get used very quickly and can be re-assigned to a different private IP/port once they finish. In very large organizations, PAT devices might have several public IPs assigned to the same interface. That gives it a pool of IP addresses to use each with 65,535 ports.

Port Address Translation was HUGE. It permitted multiple computers to all use one Public IP address. On top of that, the world never saw the internal IP address. That meant that the same private IPs could be used again on different PAT implementations. The most common Private IP ranges you are probably familiar with come from RFC 1918. Those are the ranges:

10.0.0.0/8	10.0.0.0–10.255.255.255
172.16.0.0/12	172.16.0.0–172.31.255.255
192.168.0.0/16	192.168.0.0–192.168.255.255

Most home routing equipment tends to use something in the 192.168 range so that is probably the most familiar. It is important to note that there are other private addressing spaces set aside that do not route across the Internet. Some are used for documentation. If you write a book about networking, you should use those. Some are set aside for testing, ISP purposes, or even small inter networking links. You can find out more by searching Wikipedia for “Reserved IP addresses”[9]

How fast is the network? - Latency

How fast is my network? Most folks when they hear that question think of bandwidth. Internet Service Providers (ISPs) have worked hard over the years to sell that idea.

Bandwidth = a measure of **how much** data comes in a given time.
Latency = is a measure of **how fast** it takes to get from one point to another.

If you send a sports car and a rented truck across the country, the car may get there faster. That is latency.
If they both get there, one may hold more than the other, that is bandwidth.

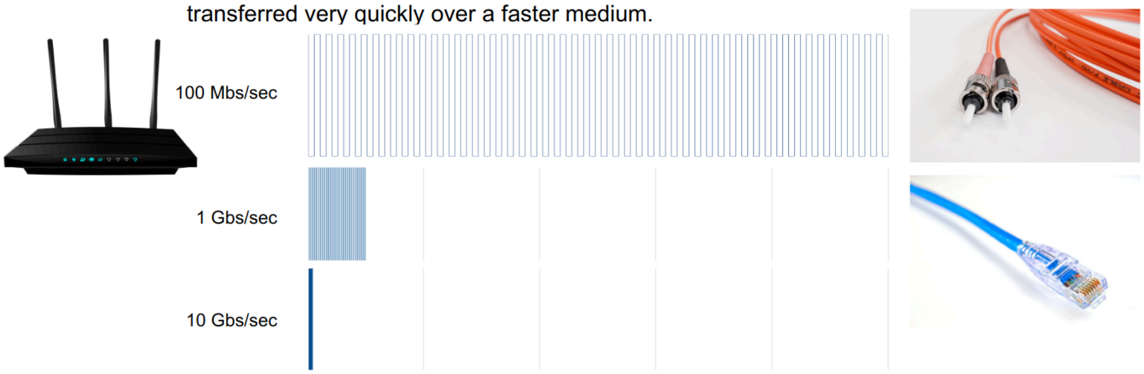
When it comes to networking, latency is heavily dependent upon your media. Different media (copper, fiber, ethernet, cable, dsl) can all have different frequencies at which they can transmit data. Even when you have the same media sometimes, it might have capabilities to negotiate different frequencies. You have probably noticed that copper ethernet has improved over the years. It used to only do 10 Megabits per second (Mbs). Then it improved to 100, and then 1 Gigabits per second (Gbs), and now 10 Gbs is possible. Wonderful, but what does that mean?

[9] "Reserved IP addresses - Wikipedia." https://en.wikipedia.org/wiki/Reserved_IP_addresses. Accessed 9 Jul. 2023.

Let's talk a little about "Latency"

Latency is measured in time. It is how fast it take to get from point A to B.

Depending on the speed of the medium, the same amount of bits can be transferred very quickly over a faster medium.

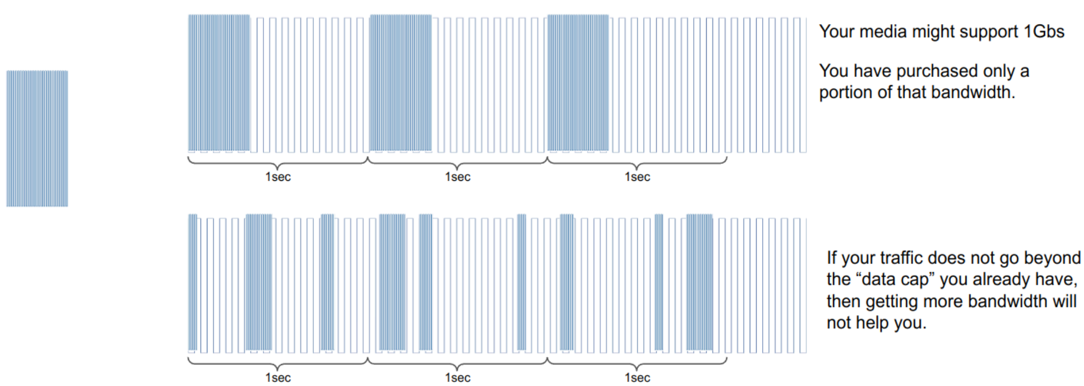


If parts of your network are traveling over slower media, guess what, you are not going to get a faster transmission than the slowest link. Latency is very important to gamers. Probably the most common way to measure is by "ping times".

What that means is in the case of 100 Mbs, in any given second you can send 100 Megabits. If you increase the frequency to 1 Gbs, then because of the increased frequency, you can send that same amount of data in a 10th of the time. In terms of latency, that data is transferred a lot faster. Switches and routers work by a mode called "store and forward". That means they wait for the whole packet to be received before they send it on to the next switch/router. So if that complete packet is transferred in a tenth of the time, it will be sent on a lot sooner than something that came in at a slower frequency. That is not what many service providers call speed however.

Bandwidth is how much data a link can handle

Bandwidth is often confused as a measure of "speed". This is a concept that has sold service packages for years.



What is the behaviour of your data? Is it small packets that need to move quickly (games, VOIP) or a continual stream (downloads, 4k video)?

How is your bandwidth shared? Packets still have to take turns on the line. (cable, DSL, fiber)

When providers talk about bandwidth as speed, it may or may not meet a customer's expectations. Whatever your network connects to, the media does not change. If your ISP has a 1 Gbs offering, chances are your latency is operating with that quality of "speed". That does not mean that is the bandwidth you are allowed to use. This is the basic concept, if you have a bandwidth package of say 250 Mbs, what that means is you are limited to that bandwidth. The medium might allow a full Gbs but once you have exceeded 250 Mb in a given second, you must wait till the end of that second before you are permitted to send more data. If you are doing a major download, then once you fill your 250 Mb in the next second, you have to wait again. This is a simplification, but it communicates the concept.

If instead you are just gaming, you are usually only using very small packets. You probably don't use more than 7 Mb in any given second. Your interest is simply in **how fast** your data is sent and gets back to you. In that case you are not limited if you do not hit your cap. You can send data any time during that second and have it take advantage of the medium's full latency. Here is where providers would like you to see bandwidth = speed. If you are not hitting your cap, then getting a larger bandwidth deal is not going to help you in your gaming. If you have other devices on your network running at the same time, then it is fully possible that you will hit your cap. It all depends upon how you are using your data. Services like streaming 4k video might see a big difference in plans. Then it is not a matter of how fast it contacts the servers. Video will usually buffer data. Whatever you see is already several seconds behind what you are downloading. If it hits a slow patch, that is not even usually noticed unless it pauses long enough before the data feed can catch up. That is a situation where bandwidth matters. That is based upon **how much** data you can get in a given time.

Even with your specific plan, whether you need speed or bandwidth, there is another issue that is common in many networking designs, that is the concept of oversubscription. You may have a great data plan, but if you are on a shared medium, like the large cable coaxial conduit running on the utility poles, you are all sharing the same medium. Then the analogy is that of an interstate highway. If there are too many cars (packets) on the road than the road can handle, everybody has to take their turn. This may not affect bandwidth for services that buffer (bandwidth), but it can have a great delay on services that can't afford to wait in a line that are latency dependent. If you are a gamer, you might find the network incredibly fast at 1am as opposed to 7pm when folks are watching TV. If networks are designed well, the core of the network is considerably faster than any of the individual nodes. That way it can handle the traffic without contention. Put another way, if the core network is 10 times faster, then it can handle ten hosts transmitting at the same time and they will experience their full bandwidth, in theory.

Throughput is where it all comes together. That is the measure of the amount of data actually getting across the network in a specific period of time.

- **Latency** - how fast the packet can travel on the medium
- **Jitter** - do the packets come in order or at the same rate? If not, that is jitter.
- **Packet loss** - do any get lost or held up along the way? Packet loss often requires packets to be resent.

- **Bandwidth** - low bandwidth can cause traffic jams. Remember Bandwidth is how **much** data can pass in a given period of time in an optimal situation. Throughput is like bandwidth but is more of a measure of what is actually happening.
- **Processing power** - Everything that handles packets needs to have the processing power available to keep everything moving the way it should. This can mean workstations, servers, firewalls, routers, switches, intrusion protection/detection systems, and more. This is often overlooked when there are discussions about networking. Newer systems tend to be faster, have more memory for buffers (helping with traffic jams), and have more features to accommodate newer methods.

Sidetrack:

One of the legends in the computing industry was Grace Hopper[10]. She was an American computer scientist, mathematician, and United States Navy rear admiral who made incredible contributions to the computing field. I highly encourage you to do some searching, and get acquainted with just how much of a huge contribution she made. When she lectured, she also had a fantastic sense of humor with a completely deadpan delivery. I invite you to check out this portion of one of her lectures that describes what is a nanosecond.
Grace Hopper Lecture [11] The rest of her lecture is fun too.

---

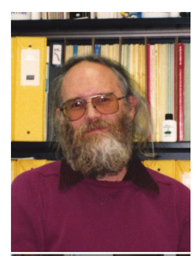
[10] "Grace Hopper - Wikipedia." https://en.wikipedia.org/wiki/Grace_Hopper. Accessed 6 Aug. 2023.
[11] "Grace Hopper Lecture - YouTube." 6 Dec. 2016, https://www.youtube.com/watch?v=ZR0ujwlvbkQ&. Accessed 6 Aug. 2023.

Part 4
Subnetting

If you are reading this, then the assumption is that you have already learned IP subnetting in a previous class. It is possible that you are using this document alongside a standard textbook. Either way, please understand that what I am about to share with you is not a replacement for the standard methods. It is primarily meant to be a reminder, or simply a different way of approaching the same concepts. You will find this information in pretty much any text you read on the subject, I am just emphasizing the concepts a little more. So, let me see if I can make it a little more fun. Ready? Here we go.

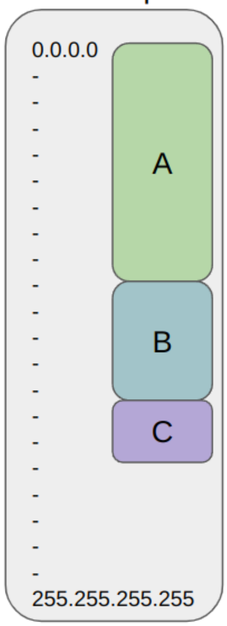
Whenever you start to think about IP subnets, it is always helpful to remember the core of what we are trying to accomplish. I find a little history helps to share the reasoning of how it came to be. Let's start by going back to the first part of this document. Remember that in the early days of the ARPANET the whole concept was about helping computers to connect. It was a fast growing community, but the craziness of what is now the world wide web was not even considered. Computers were really expensive, and the big applications of the day were telnet and FTP. It was all about getting a remote terminal or transferring files. Not exactly the thrill of modern gaming, or streaming video. It was an academic network for the most part. So when an institution wanted to connect, the first thought was, "Okay, how many hosts are you planning to have?". Contact IANA, send in the paperwork, make your case for how many IP addresses you need, and then you can get your allotment. It was simply a matter of those in charge of the routing to put values in their routing tables to tell the rest of the network, to get to that allotment of numbers, go there.

It is important to understand, all we are doing is taking a group of IP numbers, and assigning them. Once an institution has a grouping of numbers, then subnetting allows them to cut that allotment into smaller, or "sub" groupings. IANA basically says, "Here are your numbers, everyone knows to come to you to connect to them, so knock yourself out. Use them internally however you want."



Early in the days of the ARPANET, Jon Postel and Joyce K. Reynolds were the keepers of Internet numbering and socket numbering (ports). They had the joy of keeping track of the whole numbering scheme. Remember, it started small. In 1981, as part of a larger specification, the community came up with a method of classifying numbering in order to deal with the demand for IP allocations. Think about it. Some companies needed large groups of IPs to call their own, and some others needed smaller groups. The solution they came up with was when the Classful addressing system was born. Addresses were carved up into groups labeled A, B, and C. The class A networks were very large and reserved for the largest of institutions, Class B networks were the common "Large network" allocation, and Class C networks were the standard small block of addresses.

The IPv4 space



Class A

There were 128 Class A networks
Each was based on the first decimal octet
0. - 127.
Each had 16,777,216 addresses

Class B

There were 16,384 Class B networks
Each was based on the first two decimal octets.
128.0 - 191.255.
Each had 65,536 addresses

Class C

There were 2,097,152 Class C networks
Each was based on the first three decimal octets
192.0.0. - 223.255.255.
Each had 256 addresses

With the class system, it eventually limited flexibility. It is fairly obvious just by looking at the Class A groupings that restricting 16 million addresses to a little over 100 companies was not feasible. A few of the companies still hold their original allocation, but many have been reallocated.

Ownership in 2011			**Ownership Today**	
Address Block	Date	Registry - Purpose	Address Block	Registry -Purpose
-----	------	—-----------------------		
000/8	Sep 81	IANA - Reserved	000/8	IANA - Local Identification
001/8	Sep 81	IANA - Reserved	001/8	APNIC
002/8	Sep 81	IANA - Reserved	002/8	RIPE NCC
003/8	May 94	General Electric Company	003/8	Administered by ARIN
004/8	Dec 92	Bolt Beranek and Newman Inc.	004/8	Administered by ARIN
005/8	Jul 95	IANA - Reserved	005/8	RIPE NCC
006/8	Feb 94	Army Information Systems Center	006/8	Army Information Systems Center
007/8	Apr 95	IANA - Reserved	007/8	Administered by ARIN
008/8	Dec 92	Bolt Beranek and Newman Inc.	008/8	Administered by ARIN
009/8	Aug 92	IBM	009/8	Administered by ARIN
010/8	Jun 95	IANA - Private Use	010/8	IANA - Private Use
011/8	May 93	DoD Intel Information Systems	011/8	DoD Intel Information Systems
012/8	Jun 95	AT&T Bell Laboratories	012/8	AT&T Bell Laboratories
013/8	Sep 91	Xerox Corporation	013/8	Administered by ARIN
014/8	Jun 91	IANA - Public Data Network	014/8	APNIC
015/8	Jul 94	Hewlett-Packard Company	015/8	Administered by ARIN
016/8	Nov 94	Digital Equipment Corporation	016/8	Administered by ARIN
017/8	Jul 92	Apple Computer Inc.	017/8	Apple Computer Inc.
018/8	Jan 94	MIT	018/8	Administered by ARIN
019/8	May 95	Ford Motor Company	019/8	Ford Motor Company
020/8	Oct 94	Computer Sciences Corporation	020/8	Administered by ARIN
021/8	Jul 91	DDN-RVN	021/8	DDN-RVN
022/8	May 93	Defense Information Systems Agency	022/8	Defense Information Systems Agency
023/8	Jul 95	IANA - Reserved	023/8	ARIN
024/8	May 1	ARIN - Cable Block	024/8	ARIN
025/8	Jan 95	Royal Signals and Radar Establishment	025/8	Administered by RIPE NCC
026/8	May 95	Defense Information Systems Agency	026/8	Defense Information Systems

				Agency
027/8	Apr 95	IANA - Reserved	027/8	APNIC
028/8	Jul 92	DSI-North	028/8	DSI-North
029/8	Jul 91	Defense Information Systems Agency	029/8	Defense Information Systems Agency
030/8	Jul 91	Defense Information Systems Agency	030/8	Defense Information Systems Agency
031/8	Apr 99	IANA - Reserved	031/8	RIPE NCC
032/8	Jun 94	Norsk Informasjonsteknology	032/8	Administered by ARIN
033/8	Jan 91	DLA Systems Automation Center	033/8	DLA Systems Automation Center
034/8	Mar 93	Halliburton Company	034/8	Administered by ARIN
035/8	Apr 94	MERIT Computer Network	035/8	Administered by ARIN
036/8	Jul 0	IANA - Reserved	036/8	APNIC
037/8	Apr 95	IANA - Reserved	037/8	RIPE NCC
038/8	Sep 94	Performance Systems International	038/8	PSINet, Inc.
039/8	Apr 95	IANA - Reserved	039/8	APNIC
040/8	Jun 94	Eli Lily and Company	040/8	Administered by ARIN
041/8	May 95	IANA - Reserved	041/8	AFRINIC
042/8	Jul 95	IANA - Reserved	042/8	APNIC
043/8	Jan 91	Japan Inet	043/8	Administered by APNIC
044/8	Jul 92	Amateur Radio Digital Communications	044/8	Administered by ARIN
045/8	Jan 95	Interop Show Network	045/8	Administered by ARIN
046/8	Dec 92	Bolt Beranek and Newman Inc.	046/8	RIPE NCC
047/8	Jan 91	Bell-Northern Research	047/8	Administered by ARIN
048/8	May 95	Prudential Securities Inc.	048/8	Prudential Securities Inc.
049/8	May 94	Joint Technical Command	049/8	APNIC
050/8	May 94	Joint Technical Command	050/8	ARIN
051/8	Aug 94	Deparment of Social Security of UK	051/8	Administered by RIPE NCC
052/8	Dec 91	E.I. duPont de Nemours and Co., Inc.	052/8	Administered by ARIN
053/8	Oct 93	Cap Debis CCS	053/8	Daimler AG
054/8	Mar 92	Merck and Co., Inc.	054/8	Administered by ARIN
055/8	Apr 95	Boeing Computer Services	055/8	DoD Network Information Center
056/8	Jun 94	U.S. Postal Service	056/8	Administered by ARIN
057/8	May 95	SITA	057/8	Administered by RIPE NCC

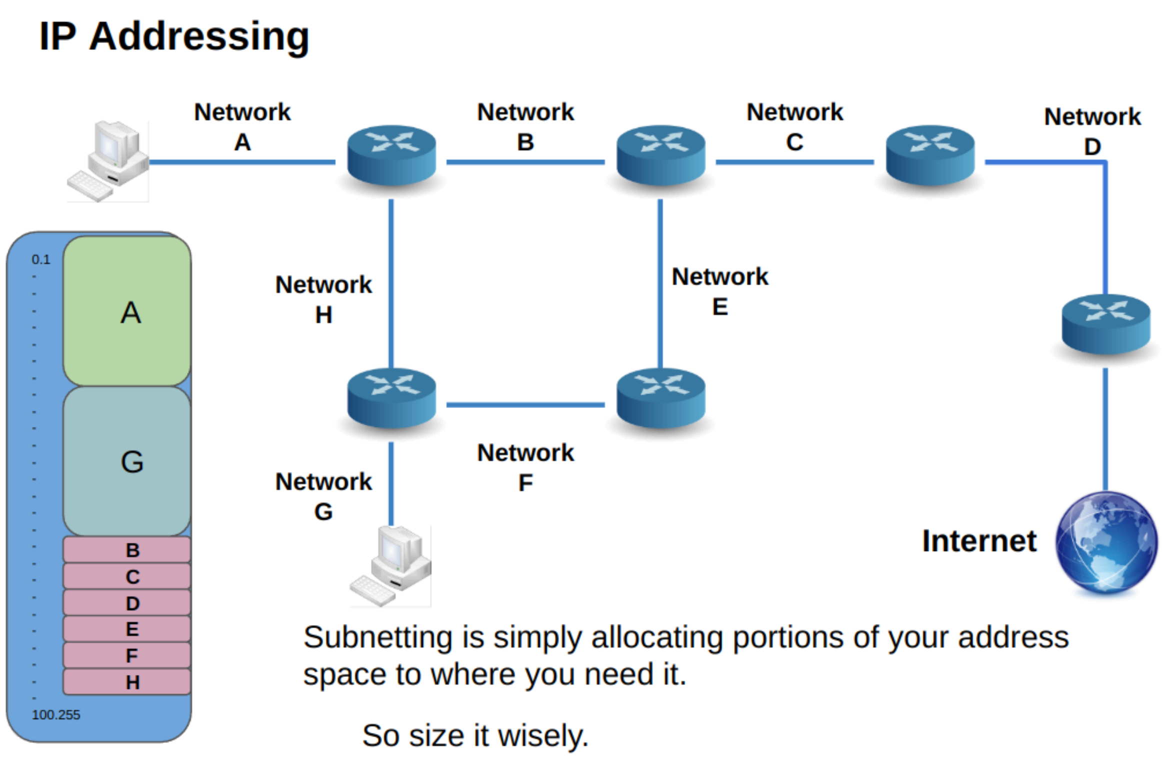
Under the class system, those were the only designations that were possible. You could not get a portion of a Class A, or a portion of a Class B, or Class C. If you owned the block you owned the whole block. Well, then the web was born, and all of a sudden there was a HUGE demand for IPs. Class C address disappeared quickly. There were none left. It was not practical to just start handing out Class B networks for those that did not need that many addresses.

The solution, …toss the system.

Now instead of a class system, we use **CIDR**, or **Classless Inter-Domain Routing**. Not very imaginative naming, but that is what you get from an engineering task force. There is some confusion left over however. Remember that the Class networks were very specific. The large networks were 0-127, the mediums were 128.0-191.255, etc. They were specific sizes as well. Today, many folks will reference a Class C, Class B or Class A as an easy way to describe the size of the network, but just know that is not accurate. It tends to confuse the issue now that networks are just described by their size and Network ID. That is what CIDR is. Don’t forget that the old system got tossed and no longer has relevance. If you remember that, then when you read something that uses the old system, you will not get confused. When you hear someone use the Class term today, they are only talking about the convenient method to describe those three specific sizes of networks. That’s it. Don’t read anything more into it then that.

Okay, well since the original rules no longer apply, what is the whole thing behind CIDR?

CIDR is simply a method for defining and then often allocating a group of IPs. When you are allocated a group of IP addresses it will probably use this notation. You will then want to divide that allocation of address into smaller groups to use in your network. To help visualize this let's take a step back.



Remember our routing diagram? We had some networks with workstations. Those networks will need more addresses than the very small networks that link routers. The linking networks really only need two addresses, one for each router. They can be really small. So, when we start to look at dividing those networks we need to think things through so that we have wizely allocated the groupings of IPs to meet our needs.

So how do we make sense of this? In my opinion, the first thing to realize is that if you try to understand IP subnetting by simply looking at the decimal, you are going to have a really hard time. I did. What that means is that we have to remember what base numbers are. We tend to think in terms of decimal. That is most likely because we have 10 fingers. If you start at zero and count up, you will go through all the digits and then when you hit ten, you have to put a marker to indicate that you have already gone through all the digits once. That is what we call 10. Go through all of them again and we have to add to the marker. That is 20. In other words we have been through the digits twice.

Computers as you probably know from class are largely a bunch of switches. They work in terms of binary. On or off. Counting in binary gets a little interesting when you only have two fingers. Your options are 0 and 1. That's it. So once you get through using both digits you have

to put that marker just like you did in decimal. To make things easier computers then work with sizes that can be easily processed through processors. To do that it is easier to work with numbers in 4 and 8 bit groups. 8 bits is a byte. 4 bits is (…are you ready for this?) a nibble. Four bits can be symbolized easily in base 16. I know, that makes for a lot of fingers. So for us humans to be able to count in base sixteen we need more digits before we hit a marker indicating that we went through all the digits. That is where the letters come in. So we count from 0-F.

IP addressing only makes sense in binary.

Remember Base Numbers?

Decimal Base 10	Binary Base 2	Hexadecimal Base 16
0	**0000**	**0**
1	**0001**	**1**
2	**0010**	**2**
3	**0011**	**3**
4	**0100**	**4**
5	**0101**	**5**
6	**0110**	**6**
7	**0111**	**7**
8	**1000**	**8**
9	**1001**	**9**
10	**1010**	**A**
11	**1011**	**B**
12	**1100**	**C**
13	**1101**	**D**
14	**1110**	**E**
15	**1111**	**F**
16	**10000**	**10**

An IP address is 32 bits. That is 4 bytes. (remember, 8 bits in a byte?) We also separate the bytes by dots. So in binary it looks like:
11111111.11111111.11111111.11111111
Okay, well eight bits in base 16 then looks like:
FF.FF.FF.FF
You can almost see that from looking at the chart above. It just gets screwy when you take that same value and put it into decimal.
255.255.255.255

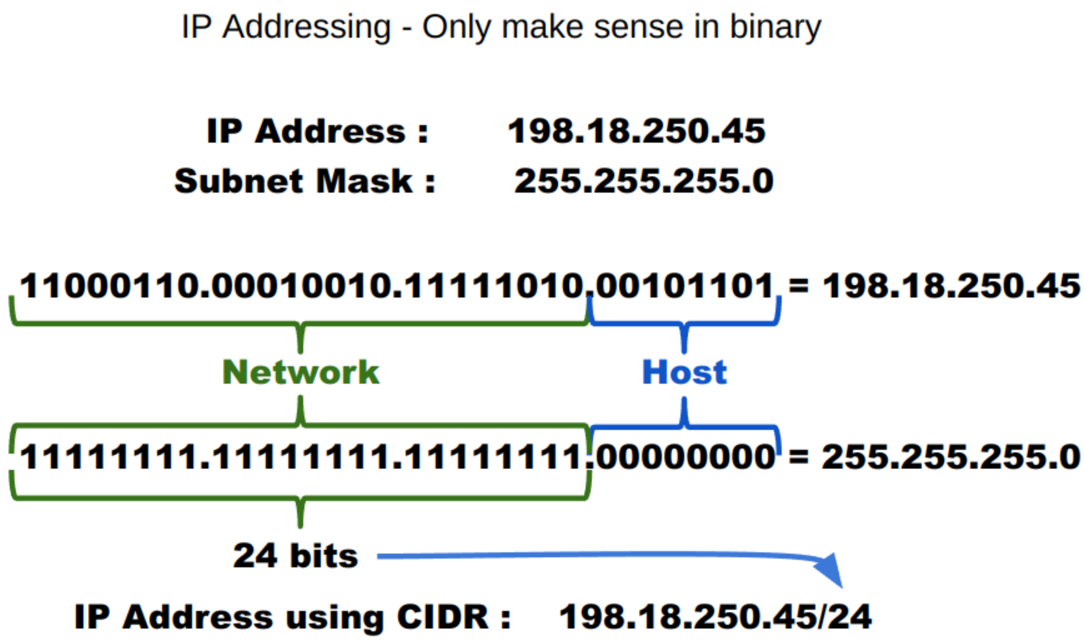
So now hopefully you can see what I mean by it only makes sense if you look at IP addressing in binary. Everything you see and know in IP is all based on the decimal equivalents of what is happening in binary.

As a reminder, for a host to work on a network it needs three things:

- **An IP Address:** Every host needs an address to communicate
- **A Subnet Mask:** A subnet mask is just to identify the size of the network. It helps the host to know what is, and is not on its network. With that, a host knows whether to send an ARP or send the packet to the Gateway.

- **The Gateway:** The gateway routes, it is where a host sends a packet if it does not know where to send it to.

With that information, then let's look at probably the most common network size used. It is a 24 bit subnet mask. In every IP address, part of the number on the left side represents the network, and the right side represents hosts on that network. In decimal, a 24 bit size network mask is 255.255.255.0. Look familiar? The advantage of this network size is that it splits the numbers in the IP address on one of the dots. The first three octets represent the network, and the last octet represents the host.



CIDR format is really just a different method of representing the subnet mask. In a subnet mask, everything that is the network is represented by a "1" bit. Everything that is a "0" represents hosts on that network. The ones will all be on the left and zeros all on the right. It is the divider. CIDR simply counts how many bits are on the left, the network.

Whenever you see a subnet mask in decimal, understand that it is simply the binary translated into decimal. **You have to think of it in binary to clearly see the division.** Now lets cover some of the basics when it comes to creating networks.

IP Addressing

Let's see if we can't make understanding subnets and variable length subnetting a little easier.

Remember: all we are doing is taking a block of IPs and separating them to meet our needs.

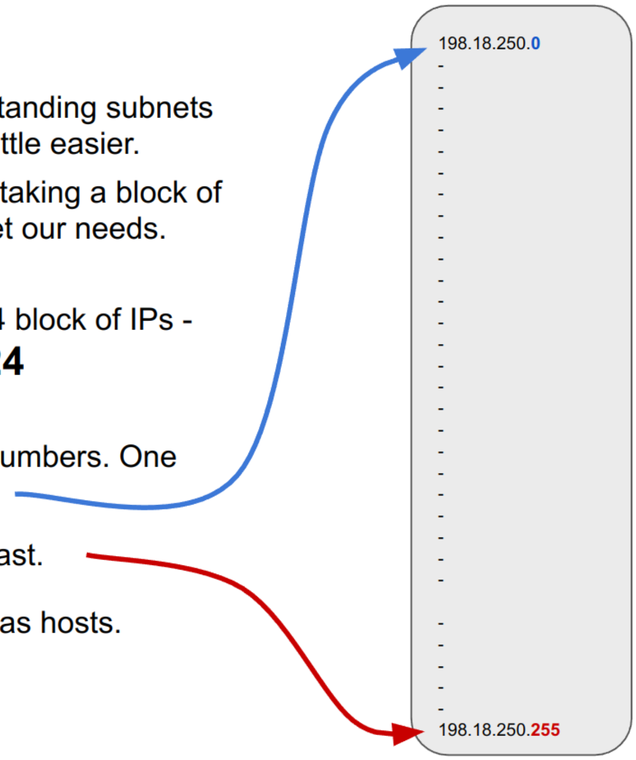
Let' pretend we were given a 24 block of IPs -

198.18.250.0/24

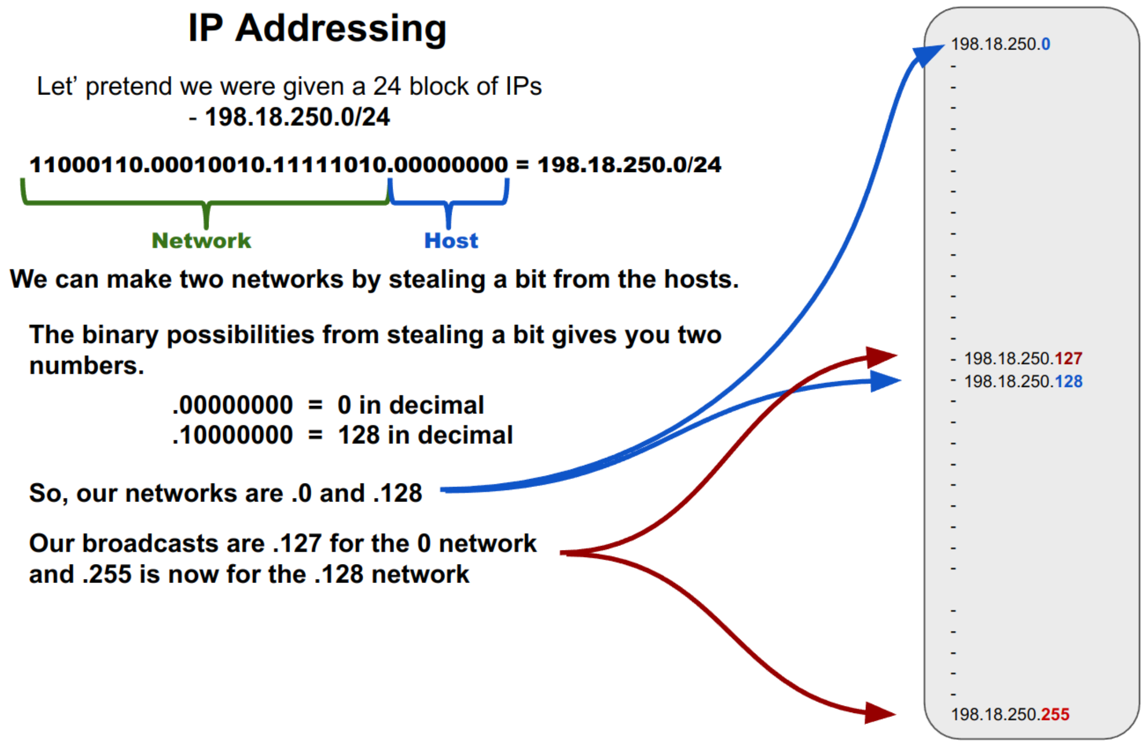
In every network you lose two numbers. One denotes the ID of the network.

The other is reserved for broadcast.

The rest of the IPs can be used as hosts.



In many books they probably covered a portion to calculate how many hosts were available for assigning IPs to in any given network. The formula is 2x– 2 where x is how many host bits. But many times they don't stress the big deal behind the -2. Every IP network needs an address to denote ***which*** network it is, i.e. the network ID, and an address for the broadcast. The broadcast address is a way to set a mac address of FF:FF:FF:FF:FF:FF. Remember that MAC addresses do not cross routing processes. Well, then how can we send a broadcast to a network from another network? If the routers/switches are configured to allow it, then they can simply send it to the broadcast address. Earlier we talked about Wake on Lan (WOL) for example. That was the technology whereby if a NIC card saw a "magic packet" with its own mac address, then it would turn on the computer for desktop management and network boot. Well, the only way to send WOL packets from a managing server on another network is to send them to the broadcast address. Remember that WOL is a protocol dependent upon Layer 2 (MAC addresses).

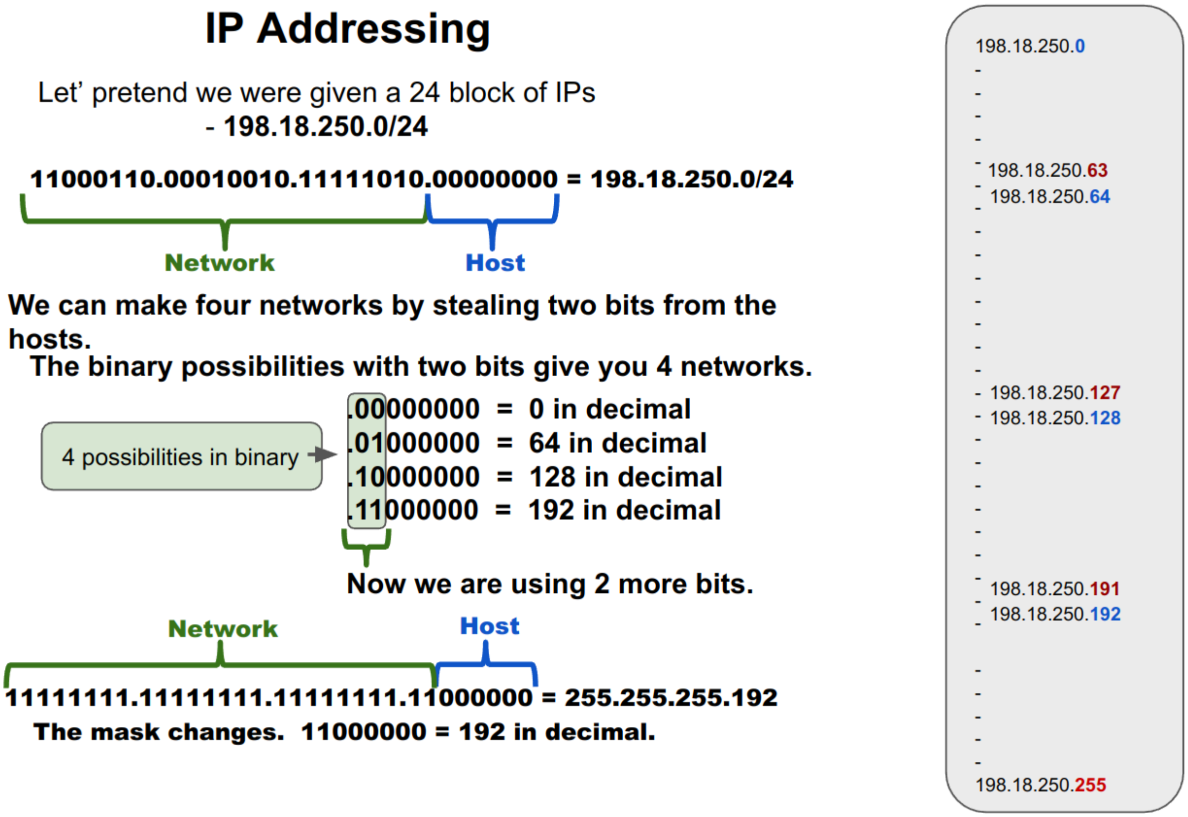


To try and make this easier, remember that we were given a full 24 bit network. If we are going to divide it, we have to take a bit from the hosts, and give it to the network side of the number. We can split up networks as much as we want this way, but the downside is we always have less hosts. So please notice. When we steal a bit to make the networks our options are 0 and 1 for the two networks. You can see it in binary as the left most bit in that last octet. It is pretty clear in binary. It is not as clear in decimal.

NOTE: once we borrowed a bit to create two networks we changed the subnet mask. We took one bit from the host side, and put it on the network side.

`11111111.11111111.11111111.10000000 = 255.255.255.128`
In CIDR we simply add a /25 to the end of the number because it is, well, now using 25 bits.

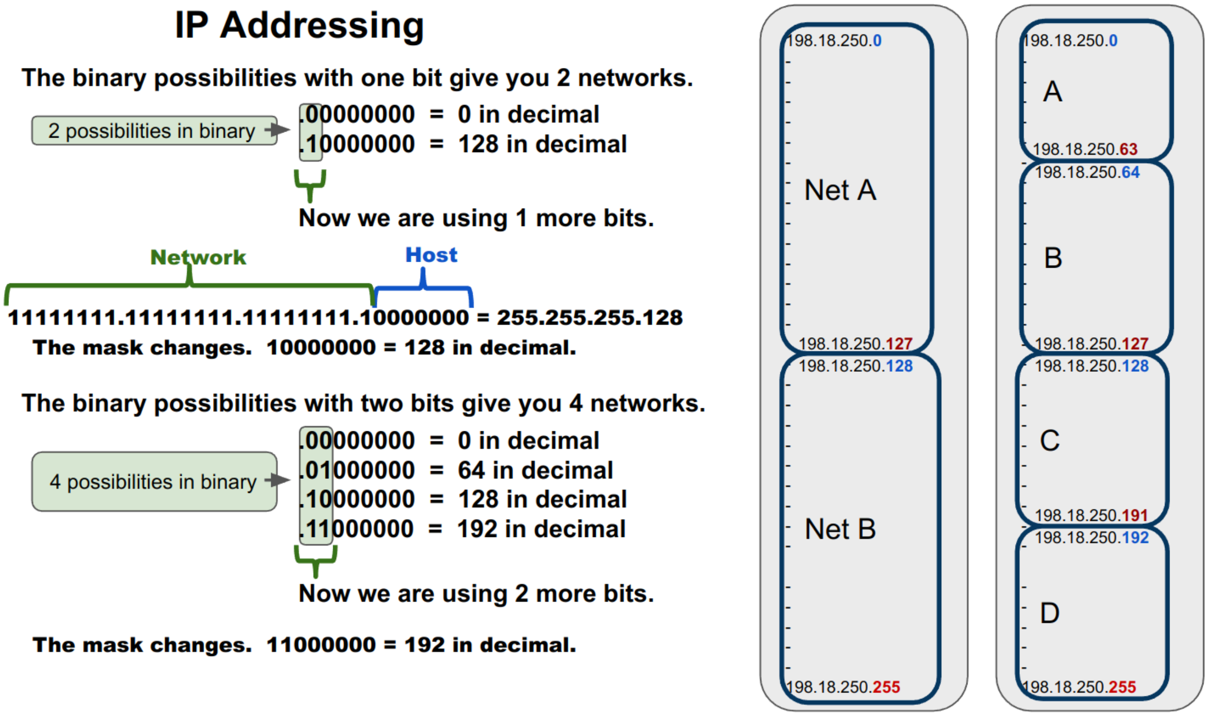
That just split our original allotment of numbers into halves. Now let’s do it in quarters.



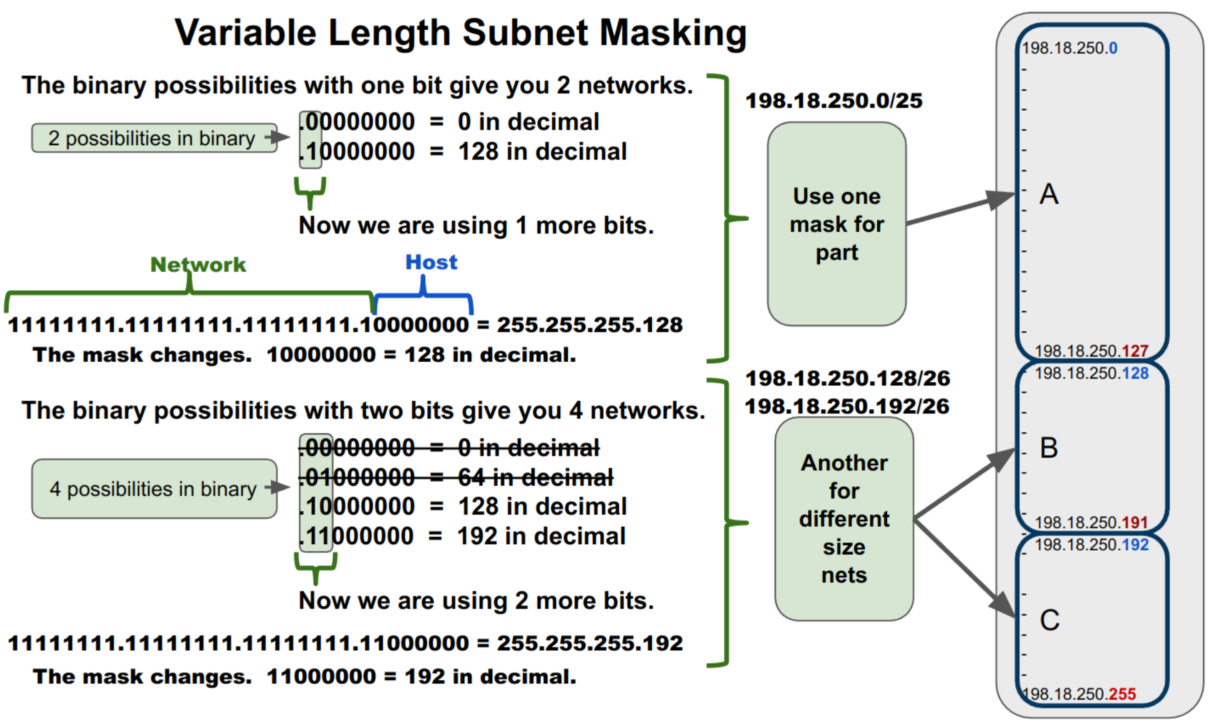
So, from the four possibilities that we can derive from the two network bits, our networks are .0 .64 .128 and .192 once you translate what is happening in binary into decimal.

Our broadcasts are .63 for the 0 network, .127 for the .64 network, .191 for the .128 network, and .255 is now for the .192 network

Notice, you need to understand it in binary and then translate it into decimal for it to make sense.



Let’s remind ourselves once again, how many networks you can make are all dependent upon how many bits you dedicate to the network portion of the number. Then it is just a matter of translating what is happening in binary to decimal. There are lots of formulas, tables and calculators that can help you in this endeavor. To understand it, go back to the binary.



Variable Length Subnet Masking (VLSM) sounds impressive, but all it really means is that all the networks do not have to be split evenly. If you follow our examples, we originally split the network allocation into two. If you only use the first half assigned to a network in your router that leaves the rest to still be allocated. As we demonstrated, you can simply create smaller networks with different subnet masks, from the remaining half. The only tricky part is, no matter how you split them up, if you allocate numbers to a network, they are no longer available. Once assigned, they will end up in your routing tables. You can't assign the same IPs in two different places in your network without breaking routing. The routers will not know which way the packets are supposed to go. If you really need the numbers, you will need to un-assign them from your network first. That is the challenge of architecting networks.

DNS

We have covered a lot so far. We have covered IP addressing, routing, sessions, history and more. With all of that, it is really clear how important IP addresses are and what is involved to get from point A to point B. What we have not yet covered is how do systems know what IP they need to go to. That is where the Domain Name System (DNS) comes in. In the early days of the Internet, for many of the applications like telnet or FTP, it was common to simply type in the IP address. This was before there were search engines. I can remember folks having little books, where they would create their own index of IPs to go to with short explanations for each entry. Teachers would go to conferences, and would take great efforts to compare notes, and provide resources of lists of IP addresses. The DNS was in operation, but for many it was difficult to get a domain name for their server. The rules were pretty strict.

I don't know about you, but I have a hard time remembering IP addresses. There is not a lot of everyday context to apply to an IP address. They are simply a group of numbers. That is what gave birth to the DNS. It provided a method to apply semi meaningful letters and numbers to a system that could then match an IP address to that resource. Originally it was just a directory. It was a text file called HOSTS.TXT that members of the ARPANET could download and place on their machines that matched hostnames to IP addresses. OSs today still have a hosts file that defines "localhost" and is a throwback to that original hosts file.

Like everything else, the Internet grew quickly, and the original hosts file grew from an ever increasing text file to a larger reference with groups of addresses being formed into domains. Several proposals for solutions were brought forth but in 1984 four UC Berkeley students wrote the first server to handle DNS named BIND. Oversight for BIND then was transferred to the Internet Systems Consortium (ISC). The overall authority for the world, the maintainers of the root servers, is IANA (Internet Assigned Numbers Authority) which is a part of ICANN (Internet Corporation for Assigned Names and Numbers)

So what is DNS?

DNS is a system of servers that maintain "zones" and their name/IP combinations. Whenever you type a name like "www.azcwr.org" it is the DNS system that will look up the name given, and provide the correct IP address. It holds more information, but we will get to that.

DNS works by the concept of “zones” a.k.a. “Domains”. If you are in charge of a zone, that means that you are in control of all the records for that zone. The most common is the address record. You get to set the name and address of any “host” in your zone. The host is the leftmost item in a “Fully Qualified Domain Name (FQDN).

www.azcwr.org

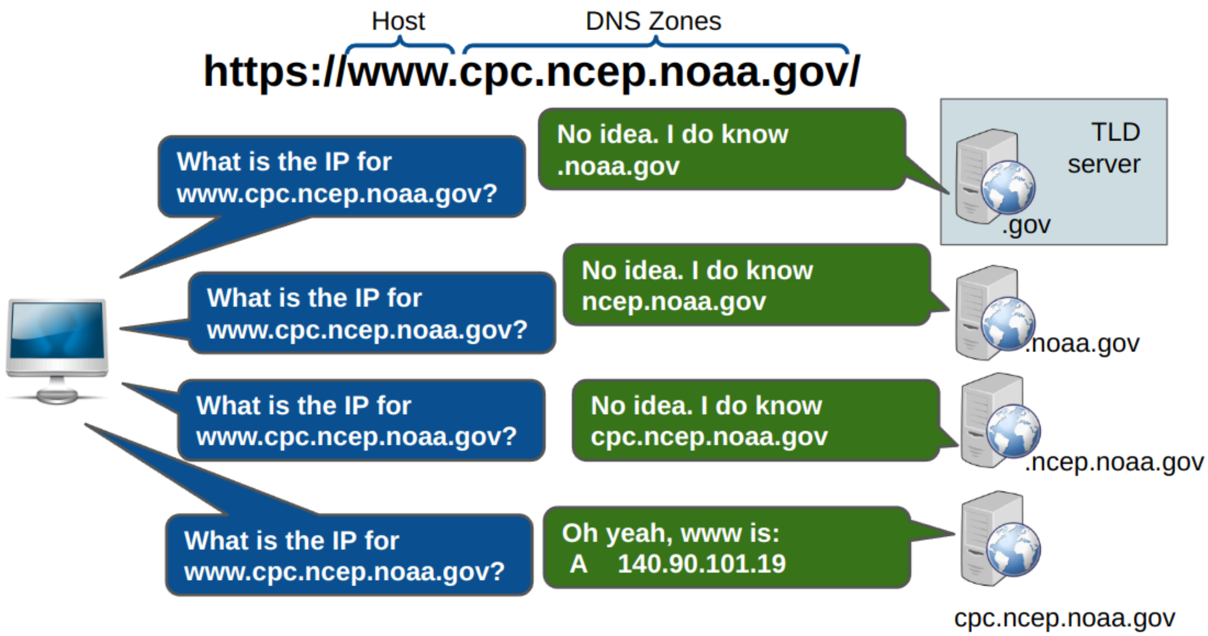
In this case, www is the actual host. That is the server you are trying to get to. The zone that holds the record is “azcwr.org”. “azcwr.org" is really a sub zone of the top level domain of “.org”. Every zone can be configured with a sub zone, but it is rare to see that. The rightmost zone however is special. It is what is called the “Top Level Domain (TLD)”. There is one more layer that is rarely seen. It is what is called the root servers. They are the built in element for all DNS related servers. They are controlled by the IANA, and are the servers that point to the TLDs. The TLDs is where the sub domains begin. Wow, that is a lot. Let’s see if we can’t make that easier.

Every zone keeps a list of the servers, etc. that are configured for that zone. It looks something like this:

```
; <<>> DiG 9.16.38 <<>> pima.tech axfr
;; global options: +cmd
pima.tech.              3600    IN      SOA     dns-itcoe. jwmccullen.pima.edu.
1641968703 3600 600 1209600 3600
pima.tech.              3600    IN      NS      dns-itcoe.
coe-ad.pima.tech.       3600    IN      NS      coe-it-ns.pima.tech.
coe-ad-ns.pima.tech.    3600    IN      A       198.18.50.103
doc.pima.tech.          3600    IN      A       198.18.48.10
fog-207.pima.tech.      3600    IN      A       198.18.60.10
helpdesk.pima.tech.     3600    IN      A       198.18.51.120
juiceshop-target-1.pima.tech. 3600 IN   A       198.18.100.111
juiceshop-target-2.pima.tech. 3600 IN   A       198.18.100.112
juiceshop-target-3.pima.tech. 3600 IN   A       198.18.100.113
juiceshop-target-4.pima.tech. 3600 IN   A       198.18.100.114
juiceshop-target-5.pima.tech. 3600 IN   A       198.18.100.115
```

This is a part of a list from our internal DNS server. Notice the names on the left, and if it is an address record, it will have an A and an IP address. This server manages pima.tech. If a host wants to know what helpdesk.pima.tech is, then it comes to this pima.tech DNS server. It is the nameserver (NS) for pima.tech. It is the Start Of Authority (SOA). If there are sub zones in the full DNS name, then there will be a record in each server pointing to the server that holds the records for that zone. Let’s go with one of those rare URLs that have several sub zones. Quite often you can see the layers of an organization, different departments, in the zone names. This often comes up if the overall governing organization has sub departments who want to manage their own hosts/servers. They might be large enough as well to have sub departments that also might have their own IT department, who want control of their own naming area and

hosts/servers. When you see the steps of a query, it is easy to see how this works. Each domain is its own server.



If a domain does not hold the record, then it needs to have the IP address and NS record of the next domain that might be in charge of the server that does.

This interchange is more an example of how DNS servers might look up a name. Typically, workstations will be configured to talk to a DNS server that handles "forward lookups". That means that you can ask it a DNS query, and it will do the work of going out and looking everything up for you. Many DNS servers are also configured to "cache" the results. That way if someone else asks for the same name in a given amount of time, the server will answer from its cache instead of going out to find the answer. This can be a big performance increase, especially for ISPs and other large forwarding entities. Most of you will probably have your workstations configured to point to your router, or modem for "DNS resolution". The router/modem will then in turn be configured to pull from an ISP or larger resource. Your DNS settings are often delivered to you through DHCP as one of the DHCP options.

Most organizations from the Internet perspective only administer their systems from the first domain, and that is why almost every record only has one domain below the top level domain. It makes it a whole lot easier for customers. Inside the organization however, there are often internal DNS servers that can't be seen from the Internet, and make use of subdomains internally.

Sidetrack:
Top Level Domains:
The first top level domains were:

.com for all commercial use. It was by far the most popular and started up a whole new business in cyber squatters. Folks would buy names that were then bought out by large businesses. Even large businesses had to spend large amounts of money to buy back the .com name that represented their business. Then there were the more generic names that have been sold for their marketability. The wikipedia "List of most expensive domain names" will give you a list of names that have sold for $3 Million or more. The top of the list is voice.com which went for $30 Million.
.**edu** is the domain name run by a non profit called Educause, and limited to postsecondary institutions and organizations that are institutionally accredited by an agency on the U.S. Department of Education's list of nationally recognized accrediting agencies.
.gov is administered by the Cybersecurity and Infrastructure Security Agency (CISA) and is reserved for US Governmental entities.
.mil is administered and reserved for the US Department of Defense.
.org was originally meant for non profit organizations, but became a sort of catch all for everything that did not fit the other original TLDs as it did not have registration restrictions.
.net was originally thought of as being for anything network and IT related. However, it was also a general purpose namespace without restrictions.

As the Internet exploded, those designations were very limiting. They quickly grew and now there are close to 1600 TLDs. The official list is controlled by IANA and can be found at: https://www.iana.org/domains/root/db.

DNS- Wait…there's more.

There is a lot to DNS. We just covered the concepts of zones, each zone being a list of DNS records, and how there is a hierarchy. We covered the basic concept of how to look up an address, but what we still need to cover are some of the other types of records that DNS holds, what they do, and how useful the information can be.

Record types: The Address record.

The basic element of DNS is to provide an IP address for a name. This is something you use every time you visit a website. The www in a URL is the host that you are translating to an IP. Many domains also have a web "default". That is why you can simply put in a domain name many times, and get the same thing that www would turn up. I have already given examples of what the A records look like in the configuration of a domain server. On a Linux/Windows/Mac host, when you want to look up a name manually, it is typically done with a client. Probably the most common by far is the utility nslookup. nslookup was the original tool issued by the ISC back when the BIND server was first developed. There was even a period where ISC was depreciating the tool, until that decision was reversed for some reason. It is the tool Microsoft adheres to in its operating system. If you are on Linux or Mac, they have gone with a tool called dig. The ISC has stopped supporting Windows and refers Windows users to use the Windows subsystem for Linux for those that want to install BIND or its tools. Both programs can be used

to pull much of the same information, but dig will often supply you with a lot more information that you simply do not get from nslookup.

By comparison, the default for nslookup will provide:
```
mcwill@box:~> nslookup www.pima.edu
Server:         203.0.113.1
Address:        203.0.113.1#53

Non-authoritative answer:
www.pima.edu    canonical name = pima.edu.
Name:   pima.edu
Address: 144.90.250.23
```

Dig on the other hand will go into more detail.

```
mcwill@box:~> dig www.pima.edu

; <<>> DiG 9.16.38 <<>> www.pima.edu
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 23855
;; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
;; QUESTION SECTION:
;www.pima.edu.                  IN      A

;; ANSWER SECTION:
www.pima.edu.           3548    IN      CNAME   pima.edu.
pima.edu.               548     IN      A       144.90.250.23

;; Query time: 0 msec
;; SERVER: 203.0.113.1#53(203.0.113.1)
;; WHEN: Sat Jul 15 15:20:25 MST 2023
;; MSG SIZE  rcvd: 79
```

The answer section of dig also presents the information closer to the format that is the equivalent to what the server configuration holds. It tends to be more informative and as far as I am concerned easier to use, so that is the tool I will be using as we go forward. If you are on Windows, you may find out that unless you install dig, which is what the ISC and other operating systems have gone to, and has the most development, you may need to use nslookup. If you are in powershell, you can also use Resolve-DnsName. It still does not provide quite what dig gives by default.

¯_(ツ)_/¯

Sidetrack:
Quick helpful things with dig:

Basic use is:

```
dig [name]
```

In my use above you may have noticed that it provides a server element. In the above lookup, it is using my router as a DNS server. One of the cool things with dig is that it is easy to direct it to another server, like Google's 8.8.8.8 for example.

```
dig @8.8.8.8[name]
```

We are about to talk about different record types that DNS uses. You can do that easily in dig by appending it right after the name. For an IPv4 address record (the default) you can use:

```
dig @8.8.8.8[name]A
```

So…

```
dig @8.8.8.8 pima.edu A

; <<>> DiG 9.16.38 <<>> @8.8.8.8 pima.edu A
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 18489
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 512
;; QUESTION SECTION:
;pima.edu.                      IN      A

;; ANSWER SECTION:
pima.edu.               600     IN      A       144.90.250.23

;; Query time: 48 msec
;; SERVER: 8.8.8.8#53(8.8.8.8)
;; WHEN: Sat Jul 15 16:30:32 MST 2023
;; MSG SIZE  rcvd: 53
```

The A record is for IPv4 addresses. IPv6 is four times as long. So they made the address record type AAAA.

```
mcwill@box:~> dig www.arizona.edu AAAA

; <<>> DiG 9.16.38 <<>> www.arizona.edu AAAA
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 44472
;; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 512
;; QUESTION SECTION:
;www.arizona.edu.               IN      AAAA
```

```
;; ANSWER SECTION:
www.arizona.edu.        60      IN      CNAME   pantheon-systems.map.fastly.net.
pantheon-systems.map.fastly.net. 30 IN  AAAA    2a04:4e42:57::645

;; Query time: 64 msec
;; SERVER: 203.0.113.1#53(203.0.113.1)
;; WHEN: Sat Jul 15 16:37:16 MST 2023
;; MSG SIZE  rcvd: 117
```

You can get both at the same time with the type ANY

```
mcwill@box:~> dig www.arizona.edu ANY

; <<>> DiG 9.16.38 <<>> www.arizona.edu ANY
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 2454
;; flags: qr rd ra; QUERY: 1, ANSWER: 3, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
;; QUESTION SECTION:
;www.arizona.edu.               IN      ANY

;; ANSWER SECTION:
www.arizona.edu.        60      IN      CNAME   pantheon-systems.map.fastly.net.
pantheon-systems.map.fastly.net. 30 IN  A       151.101.26.133
pantheon-systems.map.fastly.net. 30 IN  AAAA    2a04:4e42:6::645

;; Query time: 72 msec
;; SERVER: 203.0.113.1#53(203.0.113.1)
;; WHEN: Sat Jul 15 16:40:37 MST 2023
;; MSG SIZE  rcvd: 133
```

Notice how much information that quickly gives you. They use a canonical name, which is an alias that directs to another DNS name. Dig keeps going for you to find out what the address is of that name. In this case it is the name of the company that does their web hosting.

This especially comes into play with companies that deliver web content. They have servers hosted all over the world to try to give you the closest server. It can then provide you a different IP address of a server dependent upon what the location is of the DNS server you queried. Here is an example of asking my router, as opposed to asking Google. Notice how the names get changed dynamically, and finally identify a unique server, to finally give up the IP address of that server.

Asking my router:

```
mcwill@box:~> dig www.amazon.com

; <<>> DiG 9.16.38 <<>> www.amazon.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 19312
```

```
;; flags: qr rd ra; QUERY: 1, ANSWER: 4, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 512
;; QUESTION SECTION:
;www.amazon.com.                        IN      A

;; ANSWER SECTION:
www.amazon.com.         1367    IN      CNAME   tp.47cf2c8c9-frontier.amazon.com.
tp.47cf2c8c9-frontier.amazon.com. 6 IN  CNAME   www.amazon.com.edgekey.net.
www.amazon.com.edgekey.net. 21546 IN    CNAME   e15316.dsca.akamaiedge.net.
e15316.dsca.akamaiedge.net. 10  IN      A       23.204.249.185

;; Query time: 24 msec
;; SERVER: 203.0.113.1#53(203.0.113.1)
;; WHEN: Sat Jul 15 16:46:16 MST 2023
;; MSG SIZE  rcvd: 172
```

Now let me ask the same thing of Google. Notice the @8.8.8.8 in the command.

```
mcwill@box:~> dig @8.8.8.8 www.amazon.com

; <<>> DiG 9.16.38 <<>> @8.8.8.8 www.amazon.com
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 34034
;; flags: qr rd ra; QUERY: 1, ANSWER: 3, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 512
;; QUESTION SECTION:
;www.amazon.com.                        IN      A

;; ANSWER SECTION:
www.amazon.com.         11      IN      CNAME   tp.47cf2c8c9-frontier.amazon.com.
tp.47cf2c8c9-frontier.amazon.com. 60 IN CNAME   d3ag4hukkh62yn.cloudfront.net.
d3ag4hukkh62yn.cloudfront.net. 60 IN    A       18.155.168.220

;; Query time: 60 msec
;; SERVER: 8.8.8.8#53(8.8.8.8)
;; WHEN: Sat Jul 15 16:49:30 MST 2023
;; MSG SIZE  rcvd: 138
```

Notice the difference? Also, did you notice that to get to Amazon from my router the web page was delivered by a company called Akamai? When I asked Google, the Amazon web page was delivered by a company called Cloudfront. Wow, and most folks think they are just going to Amazon. Information from DNS can change your whole perception of how the web works. There are services that can do a DNS query for you from many sites. You might try investigating services by places like whatsmydns.net. For the query we just checked out you can see what IP you eventually would end up with from around the globe by going to:
https://www.whatsmydns.net/#A/www.amazon.com

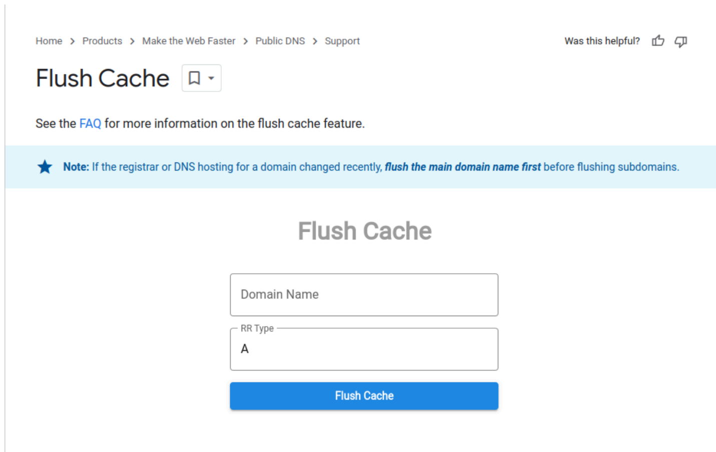
It gets wild, really fast, just by knowing a little about how DNS works. Keep this in mind if you ever find yourself having to change the IP of one of your servers. Remember I told you how DNS servers can cache queries? With services like this you can get an idea of how long it takes for your new IP to start to change across the globe. That leads me to another thing to know about DNS servers, the SOA record.

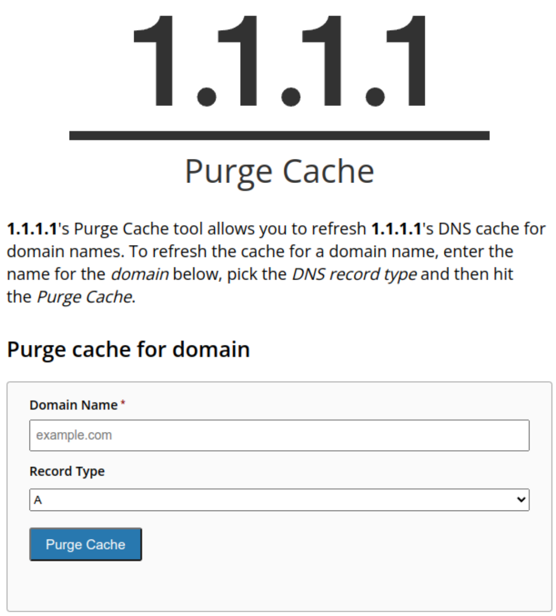
Record types: The Start of Authority.

The SOA or Start of Authority is a record that will have information that is pertinent to the specific zone. One of the more useful items is that it will contain the email address for the manager of the domain. Often that is an account that is created to route to the appropriate individuals or team. This is used primarily for any type of abuse or troubleshooting. It can also be used by law enforcement and others should there be abuses coming from your domain.

```
;; ANSWER SECTION:
arizona.edu.            7200    IN      SOA     maggie.telcom.arizona.edu.
hostmaster.arizona.edu. 2023071403 7200 3600 86400 7200
```

It will include the main name server (maggie.telcom.arizona.edu), the administrator which often has a dot replacing the "@" symbol (hostmaster@arizona.edu) and then the serial number and refresh number. The serial number is very important. It is what lets you know when the last change was made in that zone. Often they just increment by one, but many use the format here of Year,Month,Day,#of changes that day. So if you have a server www A 10.0.0.1 and you change the IP, a host out there can know that the IP address they have cached is old, and that there has been a change. The larger serial number becomes authoritative. This is why you need to be very careful with securing your DNS server. If a bad actor wants to divert your web server to their fake web server, then if they can modify your DNS server, all they have to do is set the serial number to all 9s. When you regain control of your DNS server, you then have to wait for all caches to completely timeout, because when the bad actor initiates their change, they become authoritative. There is very little you can do till ISP domain controllers expire their cache. That is why some services like Google provide an option to force clear records.





All it does is tell their servers to clear the cache for that record and ask for a refreshed record. Very handy when places like Google and Cloudflare are some of the most popular resolvers on the planet.

Record types: The mx record.

Have you ever wondered how email gets to where it needs to go when you type in somebody@google.com? How in the world is it going to get to the right servers that handle that company's email? That is also a DNS thing. That is handled by the MX record.
Did you ever wonder who handles the company email?

```
mcwill@box:~> dig pima.edu mx

; <<>> DiG 9.16.38 <<>> pima.edu mx
;; global options: +cmd
;; Got answer:
```

```
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 26756
;; flags: qr rd ra; QUERY: 1, ANSWER: 5, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
;; QUESTION SECTION:
;pima.edu.                      IN      MX

;; ANSWER SECTION:
pima.edu.               3600    IN      MX      15 alt2.aspmx.l.google.com.
pima.edu.               3600    IN      MX      20 alt3.aspmx.l.google.com.
pima.edu.               3600    IN      MX      20 alt4.aspmx.l.google.com.
pima.edu.               3600    IN      MX      10 aspmx.l.google.com.
pima.edu.               3600    IN      MX      15 alt1.aspmx.l.google.com.
```

Or, you can find out the service they are passing email through. With just a little bit of digging you can simply use the domain name and a little Googling if they are trying to disguise it.

Record types: The Text record.

Another very valuable resource is what comes through the TXT record. Text records allow you to insert, …well…text. This is the option in DNS that can be used in many ways. One of the most popular these days is to confirm validity for cloud services. The concept is that a company assumes that the customer has ownership of their DNS, and that only that customer will be able to insert records into their DNS. So what the company does is issue a name and a signing key that when their company checks for that specific key, they know it is indeed authentic. For some products that use elements in a protocol, like email for example, it can use the DNS to verify that the email did come from the source it says, and that it was legit, because of the product key in the text field. From a bad actor's point of view, this can also be a resource to point out what products they use and who they might have associations with.

```
;; ANSWER SECTION:
pifflemart.com.          3600    IN      TXT     "miro-verification=0975c0912a54ef8f4e04509f053a1c2f70ac9da4"
pifflemart.com.          3600    IN      TXT     "MS=E4F53D5B1A485B7BA06E0D36A9D38654A16609F3"
pifflemart.com.          3600    IN      TXT     "onetrust-domain-verification=cf815e1317884a00919c4ff41e7673e7"
pifflemart.com.          3600    IN      TXT     "SFMC-EcdSIWUg9PSc8V3atSL8K4HqofSpp3XYK95Gs2zE"
pifflemart.com.          3600    IN      TXT
"wrike-verification=NTMwNTkxMDphYzQ4NGU2ZDAwOWMyZTE1YmNlOGUzZWY5NmU1NjZlNzMzY2M5NTkzZjY0OGVhNDZiM2ZjZDNhZGEzZGRkN2
I0"
pifflemart.com.          3600    IN      TXT
"google-site-verification=YXLskp3Aou8VMPZyzN4ovKLB9OImvjE0Twct99j1o3c"
pifflemart.com.          3600    IN      TXT
"_globalsign-domain-verification=3ssAZMu7J08hijxtB6snFmoQ_nprFL8PfCYjJRiNBf"
pifflemart.com.          3600    IN      TXT
"_globalsign-domain-verification=BQYSCkLGqSAQK49_etRjZbpwQMRJS4fgnSdRuGOBmg"
pifflemart.com.          3600    IN      TXT
"_globalsign-domain-verification=t1aG5XXr7KpeunyzplcyPMedQSMTPyVLgCe2rOFBmD"
pifflemart.com.          3600    IN      TXT
"adobe-idp-site-verification=5800a1970527e7cc2f5394a2bfe99bcda4e5938e132c0a19139fda9bf6e30704"
pifflemart.com.          3600    IN      TXT
"adobe-idp-site-verification=7f3fb527466337ac0ac0752c569ca2ac48926dc6c6dad3636d581aa131a1cf3e"
pifflemart.com.          3600    IN      TXT     "apple-domain-verification=kId0WbgVShnay8AW"
pifflemart.com.          3600    IN      TXT     "apple-domain-verification=Q2hOt9Xg4DN4h8To"
pifflemart.com.          3600    IN      TXT     "apple-domain-verification=qF7RBX2QJfP9SvPR"
pifflemart.com.          3600    IN      TXT     "docusign=5bdc0eb1-5fb2-471c-99a0-d0d9cc5fdac8"
```

```
pifflemart.com.            3600    IN      TXT
"dtOeNuIs42WbSVe3Zf2qizxLw9LSQpFd6bWqCr166oTRIuJ9yKS+etPsGGNOvaiasQk2C6GV0/5PjT9CI2nNAg=="
pifflemart.com.            3600    IN      TXT     "facebook-domain-verification=ximom3azpca8zph4n8lu200sos1nrk"
pifflemart.com.            3600    IN      TXT
"google-site-verification=2vYJQ0zEPKkdGL8b-xDzg2alGUDx7JP-v_fqtNchRhE"
pifflemart.com.            3600    IN      TXT
"google-site-verification=4MIAIXBoB84pvAd8F45RHHYGwZO2g_Pyg5fzYrJq3dI"
pifflemart.com.            3600    IN      TXT
"google-site-verification=WnjFsU2BhWIjYZFgc3KWEt3n3PmBurN1hHCGhUwbMsk"
pifflemart.com.            3600    IN      TXT
"google-site-verification=ZZYRwyiI6QKg0jVwmdIha68vuiZlNtfAJ90msPo1i7E"
pifflemart.com.            3600    IN      TXT     "v=spf1 include:_netblocks.pifflemart.com
include:_smartcomm.pifflemart.com include:_vspf1.pifflemart.com include:_vspf2.pifflemart.com
include:_vspf3.pifflemart.com ip4:161.170.248.0/24 ip4:161.170.244.0/24 ip4:161.170.241.16/30 ip4:161.170.245.0/24
ip4:161.170.249.0/24" " ~all"
pifflemart.com.            3600    IN      TXT
"vmware-cloud-verification-c42bccc2-b759-4e91-88a9-158bea9021aa"
pifflemart.com.            3600    IN      TXT
"wrike-verification=MjUxOTk3NzpiOWIxMTRiNTNkNGZhMzdjOWNkMDg5OTgzZGNmZmVlYjdlNjhkYzQwN2U2OGI4MzdiNmM5YTIwYjEzMTUxOT
Jj"
pifflemart.com.            3600    IN      TXT
"y5rEnLEYKLwIj2mzy/lQAlFlZO9zhs2DWh/1GaMT9c2cWgeJvHiZRZqVGr2aUXOhCy7Zpd/SkXiJSFB0M0ao+Q=="
pifflemart.com.            3600    IN      TXT
"google-site-verification=lxZqOWl1JLrpFnTvJzH9wWew-Qi14ToFSug5qSjkvHs"
pifflemart.com.            3600    IN      TXT     "MS=ms72383011"
pifflemart.com.            3600    IN      TXT
"duo_sso_verification=OEGj0v4lxnMlEkwWZLG8ieSnirQtlFuZT4LWS1puxYKNHvQSOJ0KDJEdj5GMMSXv"
pifflemart.com.            3600    IN      TXT     "fastly-domain-delegation-fdd46849203012022-468492-2022-03-01"
pifflemart.com.            3600    IN      TXT     "docker-verification=c193fabe-1a62-4995-a369-bdcaa4ce798c"
pifflemart.com.            3600    IN      TXT
"cisco-ci-domain-verification=4b29fd45d5429ac519cbd0e953dfe9ef72f26d368d68850c5188e469e3f3385b"
```

I made it really small so it would fit better. But from what is listed in TXT records for example, you can get some really good guesses in this pifflemart listing:

- they probably use Cisco for their infrastructure
- they have servers in the vmware cloud
- they use Onetrust to help with compliance
- they use MS 365 and Salesforce (MS=xxxx, and SFMC-xxxx)
- Wrike for project management (a guess)
- Fastly for their DNS domain management
- Duo security for their two-factor authentication
- they use Adobe docusign to circulate and sign forms and contracts internally
- and they have Apple products which they manage through Apple Business Manager

If you have dig available, try simply entering “dig [domain] TXT” to see what you come up with. DNS queries are public and many times supply a surprising amount of information.

Another VERY important record embedded in TXT record listings are Sender Policy Framework (SPF) records. I will not go into great detail here, but SPF records are used to help battle email spoofing. It is a list of IP addresses that are permitted to send email from the domain. You are only permitted to put so many IPs in a single DNS record so you will also see “include:” statements as well. When you dig those addresses as text records, you will see another grouping of IPs, and maybe even include statements. Many times companies will hire marketing firms, or tracking solutions, that will send email on their behalf. The more companies you work with the greater the number of SPF records. DNS can also include another tool to combat

spoofed email. That is DKIM. If you check the original headers in an email you will find a statement that looks like:

```
DKIM-Signature: v=1; a=rsa-sha256; c=relaxed/relaxed; d=cymru.com;
s=dragonnewsbytes; t=1689328035;
bh=ntsX7AZDVjKY+hgkcTlzQS7ic+CsO8W4MXvwuPTWdXI=; h=Date:From:To:Subject:From;
b=TTcBBoDBE6fgx15QVewFlCatE2WvT+3VajHC6cvgFChdv62kESKUDTeGMBw1rRXVq
        7AofjDuc+IDjnM0eQoaGKGVbEiIkLbjyZargDpAwWLuM+o9o8b4gefwzm8vZrNJbIF
        XubNy1e+E3roYB0jKKAGOVJ0UmfljJzgl9sxd/tXq0suyPSLshiQlx0aNTiftsW3pC
        w2MHDbA+7hxYzhfySFXkGVYlrKEQYGXCNden5GAVSNow84b12OvSLn17WDhJcb3tX/
        fld1bhT63+xg6rnelkN6Y4ZcAPvSQO1u6oM8peFtsN39kvnWF326A5yfrT5DaoKxfp
        ntLkO4LOFlVlQ==
```

That provides the public key for digital signing. You can see it by using the s= and d= values. This is for a service provided by cymru.com. Their signature is confirmed with the dkim selector. The format is [selector]._domainkey.[domain name]. For this example to look up the key would be:

```
dig dragonnewsbytes._domainkey.cymru.com txt

; <<>> DiG 9.16.38 <<>> dragonnewsbytes._domainkey.cymru.com txt
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 1362
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
;; QUESTION SECTION:
;dragonnewsbytes._domainkey.cymru.com. IN TXT

;; ANSWER SECTION:
dragonnewsbytes._domainkey.cymru.com. 3600 IN TXT "v=DKIM1; k=rsa;
p=MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEApCAd+fR3Azpc+nP7tXMIeeQx/hPTEmAeWttGB7qg
WrkyX9iccXdg3Bh2OM4H1o62VjABR+za2FPVbE32ha02DZ+tssfEuUO+IpIMtmIe1W4yWS9qpLLcOiMmTyb2Bw
OtVCbr/AAfeHwtrvPXt7vWzulQj6rZmgMFzmW91ymgOj9TT1U6zbs5kl2Hw4wVYkvlZ"
"X1YvD+HTzgVwxFDrmUoVi2Ju/ilv+F//ed+yWkZ0XYsw9Zup4AwaUBO7oGdEM4G/w07RxC4dE7LfkbDLMYFij
Nf2YkfNIux6ohinEgXU07Jv+A1a1ehb/7Tk+tryxnsiuUIT9wif69V1cYmF6w5uQIDAQAB"
```

If you want to explore more, then google DKIM, SPF and DMARC. You will learn a lot about how email is protected these days.

Record types: The PTR record.

The PTR record is very unique. It is actually used in a very different way. So far what we have covered are forward lookups. That is a lookup of DNS names to IP. That is very useful, but what if you find you need to go in the opposite direction. You have an IP address but need the DNS name. Just like DNS has a structure for domains, there is a structure for IP numbering. They have their own zones. It takes the form of the address bytes in reverse and adds “in-addr.arpa”.

For example, to forward lookup google:

```
dig www.google.com

; <<>> DiG 9.16.38 <<>> www.google.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 17903
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
;; QUESTION SECTION:
;www.google.com.                        IN      A

;; ANSWER SECTION:
www.google.com.         48      IN      A       172.217.12.132
```

For a reverse, you use “dig -x”

```
dig -x 172.217.12.132

; <<>> DiG 9.16.38 <<>> -x 172.217.12.132
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 12308
;; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 512
;; QUESTION SECTION:
;132.12.217.172.in-addr.arpa.   IN      PTR

;; ANSWER SECTION:
132.12.217.172.in-addr.arpa. 43200 IN   PTR     lga34s19-in-f4.1e100.net.
132.12.217.172.in-addr.arpa. 43200 IN   PTR     lax02s27-in-f4.1e100.net.
```

The reverse will not necessarily be the same as the forward. Especially in instances where the forward name is spread out amongst a provider network. As we have seen, the forward IP might be changing depending upon where you are asking. But the hosts they are directed to probably have static names that will remain the same.

```
dig lga34s19-in-f4.1e100.net

; <<>> DiG 9.16.38 <<>> lga34s19-in-f4.1e100.net
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 9632
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 1232
;; QUESTION SECTION:
;lga34s19-in-f4.1e100.net.      IN      A

;; ANSWER SECTION:
lga34s19-in-f4.1e100.net. 3600  IN      A       172.217.12.132
```

There is so much more to DNS than is popularly known. I have just tried to give you some of the introductory basics, which is the scope of this document. There are elements to it like DNSSEC, where lookups can be cryptographically verified through digital signatures from zone to zone. DNSSEC is a rich topic on its own. Then there are also many more fields than what I have described here. You can get a listing on wikipedia: https://en.wikipedia.org/wiki/List_of_DNS_record_types.

IPv6 - It is not as bad as you might think

As you have probably learned from previous classes and this document, the Internet has grown to a point that we have run out of IPv4 addresses. Now, there is a big business thriving on purchasing old IPv4 space for resale. However, we do have a solution. This is not a new problem and engineers and the IT community have been preparing for it for quite some time. The solution of course is IPv6. IPv6 addresses are four times the number of bits than an IPv4 address. This makes the number of permutations absolutely phenomenal. We will not run out of IPv6 addresses for a long, long, time. We will also see how having addresses this large have given more leeway to develop standards that make working with networks a whole lot easier. However, I would like to point out, before we get into this, the only thing that changes is the address. TCP/UDP/ICMP, sessions, basic routing concepts, most of what we have covered so far, all remains the same. It is simply the address that has changed for the most part.

Another big change from what you might have gotten used to is that with IPv6, there is no need for the basic uses of NAT. Remember that NAT was primarily used to provide a method of making use of private IP space internally, so that traffic could be converted to one, or just a few, public IPv4 addresses externally. It was largely a fix for an ugly problem. Well, now that IPv6 provides enough addresses, there is no longer a need for NAT. This in turn helps to decrease latency. Every time you can decrease the amount that packets have to be processed through extra computing cycles, it only helps. If you are a gamer, IPv6 is definitely a step in the right direction.

IPv6 - <Gulp!>

IPv4 **203.0.113.0** = 32 bits expressed in decimal

IPv6 **2602:41ad:81a0:6700:0200:36ff:fed0:0dad** = 128 bits in hexadecimal
That is over 340 Undecillion addresses (10^{36})

In Binary **0010011000000010:0100000110101101:1000000110100000:**
0110011100000000:0000001000000000:0011011011111111:
1111111011010000:0000110110101101

Decimal **38.2.65.173.129.160.103.0.2.0.54.255.254.208.13.173**

The IPv4 address in hexadecimal, and using the same formatting is: **CB00:7100**
Notice: that is ¼th of the IPv6 address hex digits.

The entire IPv4 space fits in ¼th of the IPv6 numbering scheme.

First, after just getting a grasp on IPv4, IPv6 can seem overwhelming. It is not as bad as it might seem. Let's break down the concepts one by one. You will find it is not that bad, … except when you need to read an address to someone else.

If you are having the same thoughts I did when I was first reading about IPv6 you might be thinking, "Oh no! I just learned about all that subnet masking in IPv4. It was hard breaking all that down into binary and then making sense of it in decimal, now how hard is it going to be with subnets in hexadecimal?"

Well, breathe easy. When the engineers started to work on this, it was not long till they came up with a new standard. They decided that it would be a good idea to standardize the length of the Interface Identifier (IID). Huh? IID? That is just a different way of identifying the host. Remember that in an IPv4 address, the left part of the address identifies the network, and the right part of it is the host, or now the Interface ID (IID). It is the job of the subnet mask to determine where that divider is.

To make life easier, the IPv6 community has settled on simply always splitting it right down the middle with a /64 subnet mask.

2602:41ad:81a0:6700:0200:36ff:fed0:0dad

Prefix (The network ID) Interface Identifier (IID)

Understand that the normal rules of subnetting that we learned in IPv4 still work in IPv6. It is simply that the community has decided to not do that anymore except under more extreme situations. Usually the only time when you find anything off of the /64 norm is to solve very unique problems.

Why did they do this? Just like anything else, you can find the details in RFCs. Fortunately this one was drafted more like a discussion and is not quite as dry. Some of their main reasoning is:

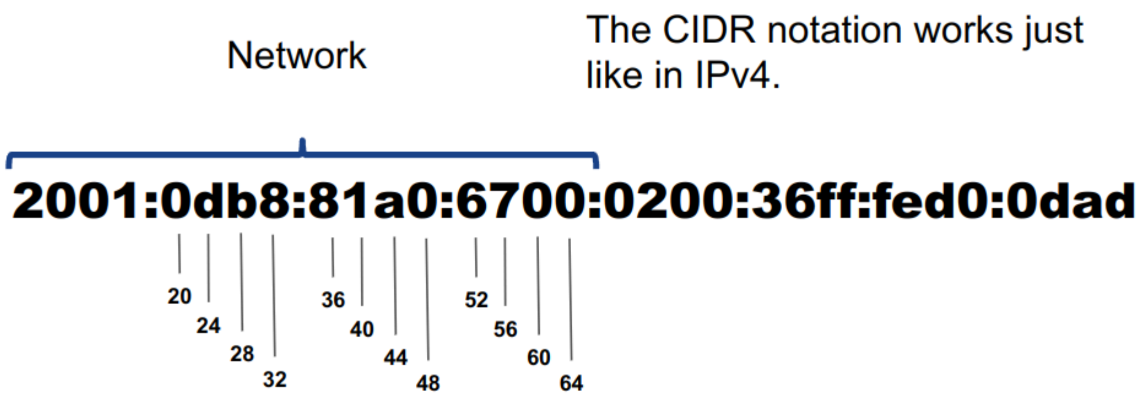
A quote from RFC 7421[12]:

> *o Everything is the same. Compared to IPv4, there is no more calculating leaf subnet sizes, no more juggling between subnets, and fewer consequent errors. Network design is therefore simpler and much more straightforward. This is of importance for all types of networks -- enterprise, campus, small office, or home networks -- and for all types of operator, from professional to consumer.*
>
> *o Adding a subnet is easy -- just take another /64 from the pool. No estimates, calculations, consideration, or judgement is needed.*
>
> *o Router configurations are homogeneous and easier to understand.*
>
> *o Documentation is easier to write and easier to read; training is easier.*
>
> *The remainder of this document describes arguments that have been made against the current fixed IID length and analyzes the effects of a possible change. However, the consensus of the IETF is that the benefits of keeping the length fixed at 64 bits and the practical difficulties of changing it outweigh the arguments for change.*

Another advantage that they did not talk about in the RFC is how this helps routing across the Internet. If the network portion of the address is uniform, then there is less calculation that needs to be applied by routers to figure out where the packet goes. Routers can then simply ignore the IID portion of the address. They really don't care. All they really care about is where the destination network is so they can send the packet on to the next hop. The structure of the IPv6 header is designed, streamlined, to put the information routers need first, so that routers don't even need to process the rest of the packet, they just send it on. In IPv4 there was more information placed before the address that was not needed to pass the packet on. As such, IPv4 routers need to spend more CPU cycles before sending the packet to the next hop.

For routers, the important part is what is called **route aggregation**. When you standardize the network portion, then when you see routing tables that use prefixes less than 64 bits, then what they are referring to is a way, just like IPv4, to direct to a whole bunch of networks with one routing statement. The advantage of IPv6 is, you now also know just how many networks that block of networks represents. On top of that, when IP networks are handed out, they are usually given network sizes that fall evenly on a digit.

[12] "RFC 7421." https://www.rfc-editor.org/rfc/rfc7421.txt. Accessed 23 Jul. 2023.



So for example. If you have a /64 network you have only one network. If you are handed a /60 network block, then that means that you have a full digit to identify your networks. With one digit in hexadecimal, that is 0-f. That gives you sixteen networks. The next would be a /56. With two digits that is 16 x 16 which gives you 256 networks. To pass on the idea of just how large the IPv6 options are, when you buy IP allocations from ARIN, for a small organization you start with a /48 address. That gives you the same amount of networks that an IPv4 Class B had in IPs. That is 65,536 NETWORKS! That is the small organization allotment.
Quoted from ARIN:

> *Organizations meeting at least one of the above are eligible to receive the minimum initial assignment of a /48.*
>
> *If requesting a block larger than the minimum assignment, provide documentation justifying the need for additional subnets based on the number of sites in the organization's network within 12 months. Your initial assignment size will be determined by the number of justified sites your network has, for example:*
>
> *2 - 12 sites: /44*
> *13 - 192 sites: /40*
> *193 - 3,072 sites: /36*
> *3,073 - 49,152 sites: /32*

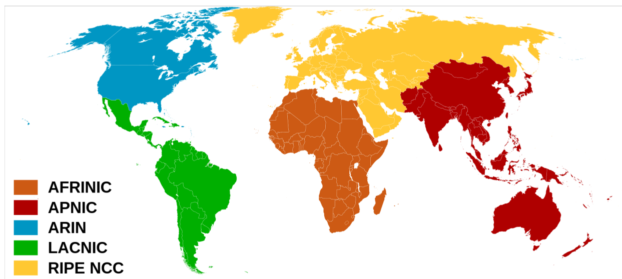
Think about it. By their policy, if you are an organization that is over 13 sites, you can get a /40. Please keep in mind that is over 16.7 Million networks, all for your organization to figure out how to allot in your routing. Each network can have 2 to the 64th power of hosts on that network. That's 18,446,744,073,709,551,616 hosts. That is over 18 quintillion hosts. So remember in IPv4, you can use nmap to scan all the hosts in a network. Not in IPv6. It would take years to complete a scan.

IPv6, Special Addresses

There are some other aspects of IPv6 that are a little different:
::/0 - Default route, just like 0.0.0.0/0 in IPv4. All zeros mean that you don't match anything, so then send it to the next router, the default route.

::1/128 - Loopback, just like 127.0.0.1 in IPv4. So now there is no place like ::1/128. You ping this, you are pinging your own box. In IPv4 they set aside a whole network just to signify your own host. In IPv6 they were a little more sensible and isolated it to a single address.

2000::/3 - Global Unicast Addresses (GUA) - These are the publicly routed IP addresses. This grouping represents the entire block of public IPs. That is everything from 2000:0000:0000:0000:0000:0000:0000:0000 - 3fff:ffff:ffff:ffff:ffff:ffff:ffff:ffff. Everything that routes on the Internet will start with a 2 or a 3. Believe it or not, that should take care of everything we need for a long time. Each of the different registry authorities were handed large blocks to allocate. Each registry was handed a /12 address space to hand out. That is 0.024 percent of the total address space. The registries are:



The African Network Information Centre (AFRINIC) serves Africa.
The American Registry for Internet Numbers (ARIN) serves Antarctica, Canada, parts of the Caribbean, and the United States.
The Asia Pacific Network Information Centre (APNIC) serves East Asia, Oceania, South Asia, and Southeast Asia.
The Latin America and Caribbean Network Information Centre (LACNIC) serves most of the Caribbean and all of Latin America.
The Réseaux IP Européens Network Coordination Centre (RIPE NCC) serves Europe, Central Asia, Russia, and West Asia.

What this represents is, each registry has enough address space within the GUA space to give each person on the planet a thousand networks each. And that isn't even making a dent in the GUA space. The vast majority of the IPv6 address space will never be used.

FE80::/10 - Link-local Addresses (LLA) - These are addresses that only communicate on the local broadcast domain or layer 2 network. They help to establish initial communication and are needed for neighbor discovery. Whenever you see an address that starts with FE80: that is only used for communication on your network. It is a way for hosts to have initial communication and find other resources like routers, and each other. IPv6 doesn't use ARP. It uses Name Discovery Protocol (NDP) instead which starts off with an operational IP address. This is the Link-Local Address.

FC00::/7 - Unique Local Addresses (ULA) - These are the equivalent of IPv4 privatized IP addresses. IPv6 is meant to do away with NAT but there are times when you want addresses that don't route across the Internet or are used for private or partner VPN connections. Private IP ranges were originally used to solve running out of IPv4 addresses, however, organizations have also leaned upon them for security aspects. Since Private addresses do not route across the Internet, it is very easy to use them inside a company and prevent any Internet host from connecting directly without an intentional NAT address being created at the gateway. Industry did not want to lose this functionality, so ULAs were allocated.

With those addresses in mind, here is the interesting part. It is common for an IPv6 host to have two or three addresses. Each host will have a Link Local to exchange information with its router. To talk on the Internet, a workstation will have a Global (Public) address. So all traffic destined for the Internet will use the Global address so the response can come back. Then, if a host is part of an organization that uses ULAs internally, it might have one of those as well. From a security perspective, three different addresses representing a single host can make life all the more interesting.

Part 5

The basic tools every network administrator should know

Dig

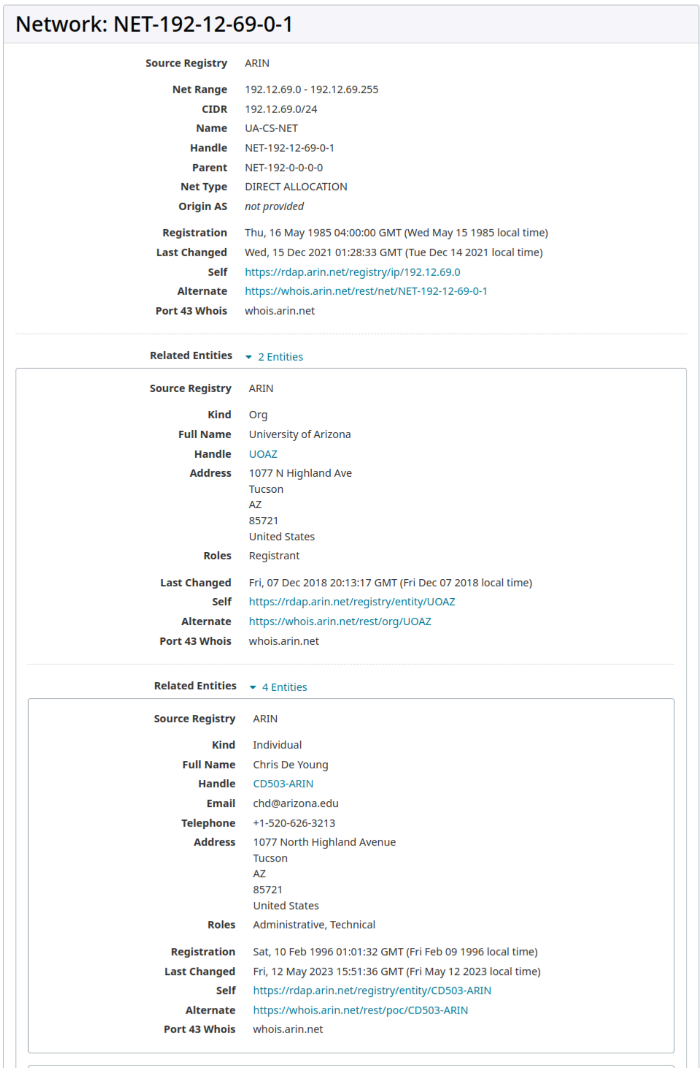
We have already done a lot with dig in the DNS section so I will not repeat what I have already shown. I would however, like to call your attention to some of the options that dig has to offer. It can be highly configurable. If the normal output is not to your liking and you would like to shorten it, “+noall +answer” will do just that. “+short” will just give you the IPs. It has these features to make it useful in scripts and reporting.

```
mcwill@box:~> dig www.amazon.com
; <<>> DiG 9.16.38 <<>> www.amazon.com
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 12047
;; flags: qr rd ra; QUERY: 1, ANSWER: 3, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 512
;; QUESTION SECTION:
;www.amazon.com.                        IN      A

;; ANSWER SECTION:
www.amazon.com.         327     IN      CNAME   tp.47cf2c8c9-frontier.amazon.com.
tp.47cf2c8c9-frontier.amazon.com. 15 IN CNAME   d3ag4hukkh62yn.cloudfront.net.
d3ag4hukkh62yn.cloudfront.net. 15 IN    A       108.138.170.85

;; Query time: 24 msec
;; SERVER: 203.0.113.1#53(203.0.113.1)
;; WHEN: Sun Jul 16 15:26:01 MST 2023
;; MSG SIZE  rcvd: 138

mcwill@box:~> dig +noall +answer www.amazon.com
www.amazon.com.         354     IN      CNAME   tp.47cf2c8c9-frontier.amazon.com.
tp.47cf2c8c9-frontier.amazon.com. 59 IN CNAME   www.amazon.com.edgekey.net.
www.amazon.com.edgekey.net. 5096 IN     CNAME   e15316.dsca.akamaiedge.net.
e15316.dsca.akamaiedge.net. 19  IN      A       23.204.249.185

mcwill@box:~> dig +noall +answer +short www.amazon.com
tp.47cf2c8c9-frontier.amazon.com.
d3ag4hukkh62yn.cloudfront.net.
108.138.170.85
```

If you would really like to see the power of some of its troubleshooting, try the “+trace” option and you can see how your lookup is resolved all the way from the root servers. Whatever options you want to be the default, you can also include them in a file in your home directory called “.digrc” and it will give you the format you like most every time.

Whois

There is also a very powerful command to have in your toolbox called whois. Many OSs will have whois tools as either a command line or app option. Probably the best version to use however is simply to visit arin.net. Whois is most effective on IP addresses. It is very commonly the first goto when you need to lookup the controlling entity behind an IP address. Let me give an example from one of the U of A addresses:



Network: NET-192-12-69-0-1

Source Registry	ARIN
Net Range	192.12.69.0 - 192.12.69.255
CIDR	192.12.69.0/24
Name	UA-CS-NET
Handle	NET-192-12-69-0-1
Parent	NET-192-0-0-0-0
Net Type	DIRECT ALLOCATION
Origin AS	*not provided*
Registration	Thu, 16 May 1985 04:00:00 GMT (Wed May 15 1985 local time)
Last Changed	Wed, 15 Dec 2021 01:28:33 GMT (Tue Dec 14 2021 local time)
Self	https://rdap.arin.net/registry/ip/192.12.69.0
Alternate	https://whois.arin.net/rest/net/NET-192-12-69-0-1
Port 43 Whois	whois.arin.net

Related Entities ▾ 2 Entities

Source Registry	ARIN
Kind	Org
Full Name	University of Arizona
Handle	UOAZ
Address	1077 N Highland Ave Tucson AZ 85721 United States
Roles	Registrant
Last Changed	Fri, 07 Dec 2018 20:13:17 GMT (Fri Dec 07 2018 local time)
Self	https://rdap.arin.net/registry/entity/UOAZ
Alternate	https://whois.arin.net/rest/org/UOAZ
Port 43 Whois	whois.arin.net

Related Entities ▾ 4 Entities

Source Registry	ARIN
Kind	Individual
Full Name	Chris De Young
Handle	CD503-ARIN
Email	chd@arizona.edu
Telephone	+1-520-626-3213
Address	1077 North Highland Avenue Tucson AZ 85721 United States
Roles	Administrative, Technical
Registration	Sat, 10 Feb 1996 01:01:32 GMT (Fri Feb 09 1996 local time)
Last Changed	Fri, 12 May 2023 15:51:36 GMT (Fri May 12 2023 local time)
Self	https://rdap.arin.net/registry/entity/CD503-ARIN
Alternate	https://whois.arin.net/rest/poc/CD503-ARIN
Port 43 Whois	whois.arin.net

Whois will often give you the networks that were allocated to the company that owns them. Notice that they report the network in a network range [beginning IP - ending IP] and they also will report it in CIDR. Most reports of networks will also include contacts. They may have both the technical contacts, and often abuse contacts. If you are maintaining a network, you may find yourself as one of these contacts. If your network is being used to attack someone else, you may get a call. If you are getting attacked, and it is from an ISP or other entity, they might be very appreciative to know that one of their assets have been compromised. It is amazing how sometimes that can grow into professional relationships and helping each other out. If you find that you are getting attacked by a larger group that you don't consider friendly, you have the

information to simply block the entire range in your firewall. When you are using ARIN, also follow the organizational links. You may find that they have many more network blocks assigned to them than you first thought.

Nmap

Nmap is an incredibly powerful tool. It can do many things. It is probably the most used tool for scanning networks and hosts. It is highly scriptable and is powerful enough to modify network packets in ways that will send unexpected behavior to elicit responses off of normal traffic. There are aspects to nmap that if used improperly, against the wrong IPs, can be quite illegal. It has also been a Hollywood goto showing up in movies like “The Matrix Reloaded", “The Listening”, “Live Free or Die Hard”, “Bourne Ultimatum” and more. If you want to learn about many of the fine details of networking it can be invaluable. Gordon “Fyordor” Lyon, the author of nmap also has a book, and is a University of Arizona graduate. One of the more valuable and common uses of nmap can help you when you are trying to get an idea of what is in an IP range. Instead of reaching out to the IPs, you can do what is called a list scan. It can do a reverse lookup of IPs in a given network range. In essence, it will do a range of “dig -x” lookups in one command.

Using the University address we used in our whois lookup, we can see:

```
nmap -sL 192.12.69.0/24 --dns-servers 1.1.1.1
Starting Nmap 7.92 ( https://nmap.org ) at 2023-07-16 16:23 MST
Nmap scan report for 192.12.69.0
Nmap scan report for 192.12.69.1
Nmap scan report for cat45k-1.cs.arizona.edu (192.12.69.2)
Nmap scan report for 192.12.69.3
Nmap scan report for raptor.cs.arizona.edu (192.12.69.4)
Nmap scan report for optima.cs.arizona.edu (192.12.69.5)
Nmap scan report for 192.12.69.6
Nmap scan report for moki.cs.arizona.edu (192.12.69.7)
Nmap scan report for zuni.cs.arizona.edu (192.12.69.8)
Nmap scan report for 192.12.69.9
Nmap scan report for atlas.cs.arizona.edu (192.12.69.10)
Nmap scan report for corellia.cs.arizona.edu (192.12.69.11)
Nmap scan report for bitmax.cs.arizona.edu (192.12.69.12)
Nmap scan report for saguaro.cs.arizona.edu (192.12.69.13)
Nmap scan report for lab-laptop.cs.arizona.edu (192.12.69.14)
Nmap scan report for leitura.cs.arizona.edu (192.12.69.15)
Nmap scan report for cluproxy.cs.arizona.edu (192.12.69.16)
Nmap scan report for boojum.cs.arizona.edu (192.12.69.17)
Nmap scan report for victoria-vm.cs.arizona.edu (192.12.69.18)
Nmap scan report for 192.12.69.19
Nmap scan report for oxford.cs.arizona.edu (192.12.69.20)
Nmap scan report for cambridge.cs.arizona.edu (192.12.69.21)
Nmap scan report for zora.cs.arizona.edu (192.12.69.22)
Nmap scan report for 192.12.69.23
Nmap scan report for 192.12.69.24
Nmap scan report for cgi-res.cs.arizona.edu (192.12.69.25)
Nmap scan report for 192.12.69.26
Nmap scan report for 192.12.69.27
```

```
.
.more examples edited down.
```

The UofA uses imaginative names for many of their hosts, but in a corporation, a scan like this can identify lots of resources simply by their DNS name. If you are getting attacked, you can use this technique to attempt to get some idea of what is taking place on that network. Many times large groups of ISP IP allotments will have default names that stand out as customer distributed IPs. If the other neighboring hosts all have DNS names that end in .cn (China) then you might have a different risk level.

Route servers

Here is a resource that is fun, and if you know how to use it can give you a completely different insight. As we covered in routing, your routers will have default routes until they find their way to the Internet. When you have routers that are participants in the Internet BGP routing tables, they have to know how to get to all the networks. Have you ever wondered what that looks like? Well there is a project that has developed route servers. Route servers are hosts, some servers, some routers, that are connected to the Internet at different junctures, simply for the ability to allow for public logins and get a perspective from different locations. This helps to check BGP routing paths to see if they are indeed routing correctly and not being hijacked. You will likely need to install telnet as it is almost never used anymore. Then check out the list at: https://www.routeservers.org/

If you connect, remember that for most routers you can get help by simply pressing "?". As you enter the first word in a command, put a space and enter "?" again to see the options. Just by connecting to one of them and issuing a "show bgp summary" I could see 15,062,728 routes in IPv4 and 2,907,072 in IPv6. You can also look up specific routes to see the next autonomous system network they would be heading to. This gives a perspective well beyond what you would get from traceroute.

Looking Glass servers

Looking Glass servers do much of the same thing. Probably the best is run by a huge Internet provider called Hurricane Electric. https://lg.he.net/ This server allows you to pick which router from their global looking glass offerings, and get incredible detail. If you select BGP route, you can then see the path for whatever destination from that perspective. BGP looking glass options and details are beyond the scope of this document, but it offers you a portal into a whole new world of network. It starts to go into much deeper levels, which unfortunately I can only lead you to at this point. You will have to start to explore on your own. If you really want to explore the topics of networking professionals, there is a wealth of information on topics that are VERY advanced in past conferences of NANOG. https://www.nanog.org/ and https://www.youtube.com/@TeamNANOG

RIPEstat

A good tool to pull many services in a one stop point is through the use of RIPEstat. There are elements like the whois information where you will need to check other services as it does have a very specific server it is pulling from. RIPE is the European numbering authority. One of the more interesting features is the BGPlay for visualizing BGP connections across the Internet. You can find it at https://stat.ripe.net/app/launchpad

CIDR as a filter

You have probably already realized this by now, but I thought I would give it special attention. It is important to have a basic grasp of CIDR. Earlier it was introduced as just a newer way of signifying a subnet mask. It was used to identify networks for routers so that you could assign subnets to different locations in your network.

I would like you to take a moment and give it a little more thought. CIDR is used for so much more. Now, I would like you to start thinking of CIDR as a method of defining the size of a block of IP addresses. CIDR is heavily used to define firewall rules. If you want to block off large groupings from China, you will be defining them with CIDR. If you are using wireshark to look at traffic coming from a particular organization, you will probably be defining your filters with CIDR. If you are monitoring a SIEM doing threat analysis. Looking at attacks and comparing them with others, you will probably be using CIDR. In fact, you can use it in most of the tools I just shared with you. Hopefully this guide has made that easier. If you are on linux or a mac there is a terrific command line tool called ipcalc, there is another that does much of the same thing with sipcalc. They are quick and effective subnet calculators. For example, take any IP range and give it a CIDR.

```
ipcalc 192.12.69.0/24 --all-info
Network:         192.12.69.0/24
Netmask:         255.255.255.0 = 24
Broadcast:       192.12.69.255
Reverse DNS:     69.12.192.in-addr.arpa.

Address space:   Internet
Address class:   Class C
HostMin:         192.12.69.1
HostMax:         192.12.69.254
Hosts/Net:       254
```

It can quickly give you the netmask, broadcast, minimum and maximum usable hosts. Remember, you lose two addresses. One is the network ID and the other is the broadcast. Notice it also tells you that in the old specification, it would have been part of a Class C allocation. The address class DOES NOT represent the size. It is also a public address.

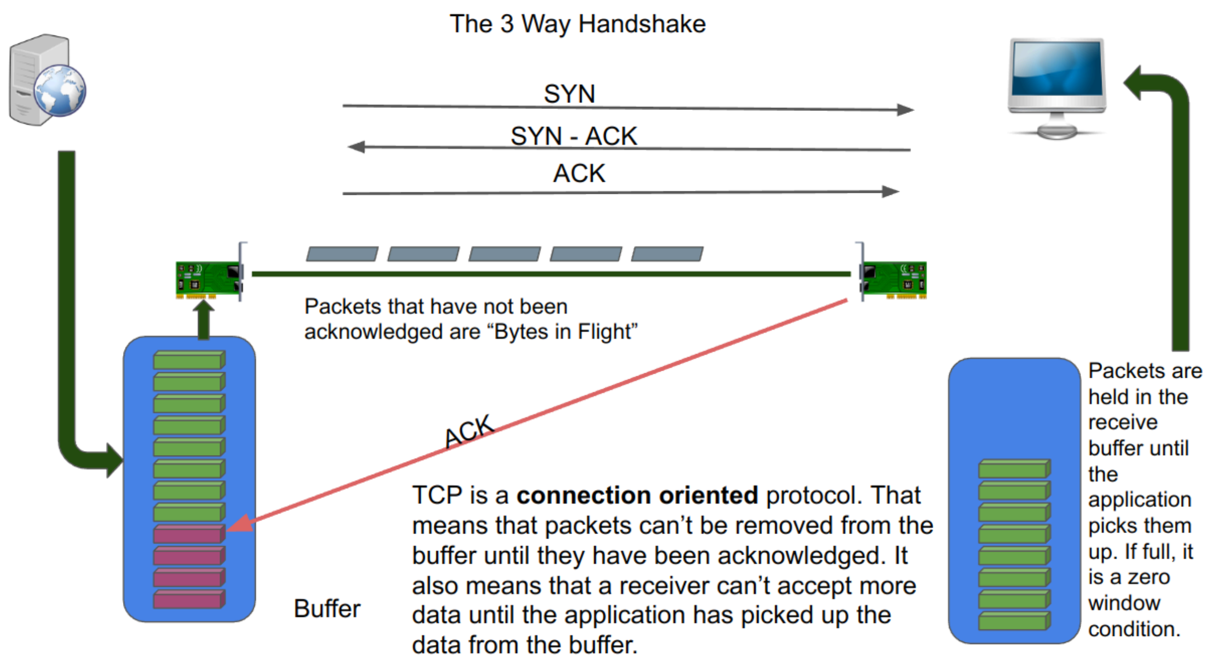
The TCP Three Way Handshake and Sessions

Just before we venture into the Transmission Control Protocol (TCP), it is important to make sure we define the difference between TCP and UDP. User Datagram Protocol (UDP) was actually created around ten years after TCP. The main difference is that TCP is a connection oriented protocol and UDP is a connectionless oriented protocol. Great. What does that mean? …glad you asked.

Back at the beginning of this document, I described how the whole concept of packet based networks was created as a means to replicate the point to point data lines that existed before the ARPANET. Taking what was a dedicated line and turning that into packets produced quite a challenge, especially in the early days when much of the infrastructure was not the most stable. Programs were written to have the same fluid communication like when we connect a USB cable to a computer. Hence, when you made a connection in a program it was called a socket, just like you connected a cable by directly plugging it into a socket. When you did this over a network, it became slightly abstracted. You were now plugging into a virtual socket, …a port. Now when we connect via TCP/IP we connect to a computer address and a port. Sometimes it helps to visualize where the concepts came from.

Back to the early days. Programs were written to use these connections. Creating a separate circuit to every computer was obviously not going to scale. This was well before the world wide web, so they had no idea just how big the Internet was going to take off. Their solution to this was to write a program that would behave like a directly connected circuit but have the ability to take the data, chop it up into smaller packets, give them addresses, and re-assemble the data on the other end. If both sides were running this program, then they could operate as if they had a separate circuit and be none the wiser. This program at its inception was first called the Transmission Control Program. It wasn't until later that it became the Transmission Control Protocol that we know today. As I quoted earlier, it was just one program, as it developed and grew more complex it became layered and renamed TCP/IP.

Now, think about the complexity of what is involved to recreate a socket. First of all, the program has to not worry about how the data gets to the other side. It just needs to know how much it can send and what is coming back. To handle that TCP creates buffers on both sides, temporary storage areas to work from as it is processing packets going and coming. Think of it like a loading dock for a company. If that gets full, you have to tell folks, "Hey, hold on a minute, we have to process what we have". In the case of data, we also have to be able to resend a packet if it gets dropped along the way. That is only really possible if we have a copy of it in our buffer/loading dock to pull from. Does that set the picture?



The process works like this:

- A conversation is started with a three way handshake. This says basically “Hey let’s talk and oh by the way can we agree on these parameters?
- Then the application sends the data meant for the other computer to the TCP buffer, the loading dock if you will. It takes the data and boxes it up in nice neat packets with to: and from: addresses. It then launches it out onto the network.
- The packets get routed over the network as we have described earlier. With TCP it demands a receipt in the form of an Acknowledgement packet (ACK). When you consider the loading dock example, copies of those packets need to be held on the dock (in the buffer) until the receipt comes in. Once that ACK (receipt) arrives, then those packets can be removed and more room can be available for more packets to send. Every packet has to be accounted for. If the process backs up, you get delays. When that happens then you have to wait until things can get flowing again. By the same token, when the buffer gets full on the receiving end, then the receiving computer has to send a signal back, “WAIT, I am not ready. My loading dock is full because the program on my end has not come by my loading dock to pick up all the packages.” It is the responsibility of the Program on the receiving side to remove data from the receiving buffer. This situation is also known as a Window full event. More on that later, just know that when we talk about windows in a TCP sense, we are talking about buffers and not the operating system. It is important to know, the main focus of TCP is to verify the connection, and make sure that every packet gets delivered. **THAT** is what is meant by a connection oriented protocol.

UDP is different. It was created around ten years later. It is a connectionless based protocol. The idea behind UDP was to increase speed. When TCP was created computers were slow,

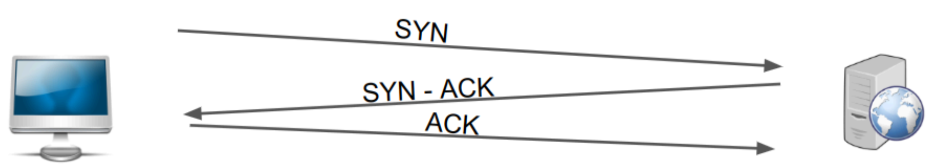
when I say slow, I mean SLOW. It is absolutely staggering how much computing has increased over the years. The ARPANET large computing machines would have been completely outclassed by today's $15 Raspberry Pi Zero 2 W. TCP was built for stability as well as performance. However, it became apparent that there was a growing number of applications that did not require the guaranteed delivery that TCP was designed to provide. Especially today, some applications have enough computing ability that they can handle the process of determining what data might need to be requested if lost. Thus UDP was born. It has only four fields: Source Port, Destination Port, Length, and Checksum. It does not worry about buffers, flow control, flags, if a packet was dropped, or if a packet was able to be received. It simply sends the packets. It is up to the application to handle how well the data is performing.

Maybe this will help:
If you are watching TV through a streaming resource, you are probably watching through TCP. Have you ever been watching and all of a sudden have it stop, and wait till it catches up? That is the principle behind TCP, a connection oriented protocol. You do not miss anything.

UDP however, if you have ever watched a video and have it pixelate but kept the video running, you are missing packets but continue anyway. That is the principle of UDP. You may experience loss but you don't worry about it. Voice phone conversation is also a UDP connectionless example. Do you get drops at times? Miss some words, but the conversation continues real time? That is a connectionless protocol in action.

New protocols are always developing. QUIC is developed by Google and is an interesting blend of TCP elements delivered over UDP. But, that is a different discussion. Let's stick to the basics right now shall we?

Options

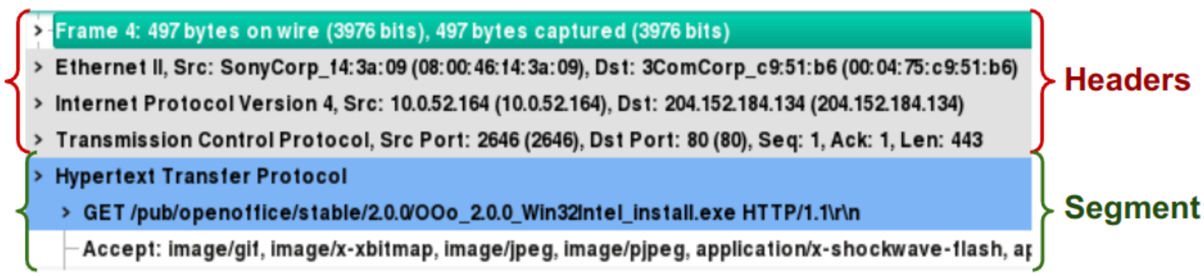


There are quite a few options that can be offered, but we will only focus on some of the primary ones. Some of the options have made it to adopted RFC status and others have not. You can find the list at www.iana.org/assignments/tcp-parameters

Options: (12 bytes), Maximum segment size, No-Operation (NOP), Window scale, No-Operation (NOP), No-Operation (NOP), SACK permitted
- Maximum segment size: 1460 bytes — **Maximum Segment Size**
- No-Operation (NOP)
- Window scale: 2 (multiply by 4) — **Windows Scale**
- No-Operation (NOP)
- No-Operation (NOP)
- TCP SACK Permitted Option: True — **Selective Acknowledgements**
  - Kind: SACK Permitted (4)
  - Length: 2

The TCP session always starts with a three way handshake. This is the opportunity to set the tone of the conversation. It starts with an initiator called a SYN (synchronize). This is the start of a stateful conversation and what initiates sessions. (remember the sessions with firewalls and network address translation?) It also sends options. It is followed by an acknowledgement (ACK) from the other side, which also sends a SYN at the same time (SYN-ACK). Then to finish it up, the initiating side acknowledges that SYN. TCP has evolved and there are options that, if agreed upon, can greatly improve sessions. Just before we cover those, I would like to bring a couple of items to your attention.

1. TCP has some options that can change throughout the conversation. Some however cannot. If you are analyzing a session with Wireshark, you always want to get the handshake in the capture. That way it provides Wireshark with more information about how the two sides are communicating.
2. The three way handshake has a unique opportunity in a capture as these three packets never have any segment information. They are just headers. Headers are the portion of the frame that has the information about how to get from point to point and what session it is. It is the segment that represents the actual data that is being sent.



In the three way handshake, since there is no segment, that is the optimum time to calculate the time difference between when the packet was sent, and when it was received. That is what is known as the path latency and is the best measurement of how fast the network really is. If you are taking the Wireshark class, we will explore this in greater detail.

Windows Scale

When TCP was developed, it set aside a receiving buffer of 65,535 bytes. With the amount of data we are sending these days, many times this is not very much. This is equivalent to the receiving loading dock. If all of a sudden you are getting a lot of freight sent to you, more than you can handle, you need to tell the sender to stop sending trucks (a zero window condition). This is the same concept with the buffer. During the handshake there is a windows scale option, where both sides can modify the size of their buffer and, to use the analogy, increase the size of their loading dock by factors. This will allow for more “bytes in flight”. Those are the bytes that are sent and awaiting an ACK, a receipt if you will, so that those packets can be cleared for more. Windows scale can be critical in preventing pauses in your network traffic that result from the other side dealing with a sluggish application. “Hey application! We have packages on the loading dock waiting for you to pick them up. I can’t accept any more until you clear them off my dock!”

Selective Acknowledgements

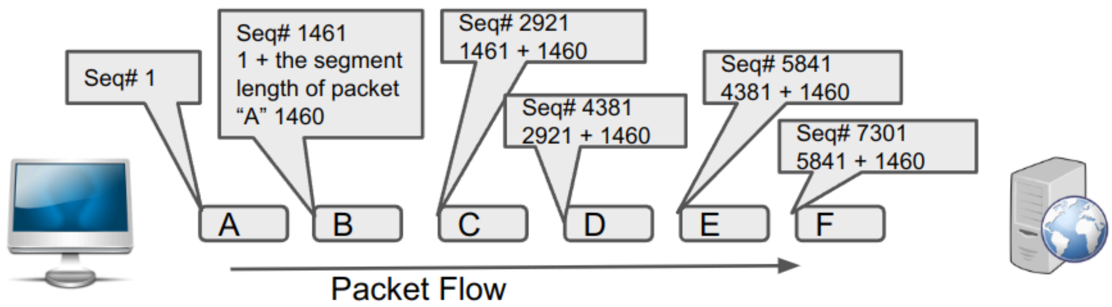
One of the major features of TCP is guaranteeing getting all the packets. To do that, each packet needs to be acknowledged, and sent again if it is dropped. This is how that works. Imagine the packets going across the wire. Here I am going to label them A-F. In this example let's imagine that packet C never makes it to the destination. The way the receiver responds is to get obnoxious by repeatedly remarking that it got the packet just before.
"I got packet B"
"I got packet B"
"I got packet B"

These are known as duplicate acknowledgements, or DUP ACKs. The specification is that after seeing three DUP ACKs ("I got packet B"), then the send goes: "Oh, I get it. You are trying to tell me that you never got packet C" and then sends packet C again. Now you will see many times that applications will react differently. Some don't want to take any chances and will send more than three DUP ACKs just to quickly get the point across and compensate for the possibility of any of the DUP ACKs being lost. The other ugly part of the original specification is that in this process, even if D, E, and F successfully made it while packet C was lost, the sender has to resend everything from C on. Now that we are dealing with larger buffer windows and can have a lot more data in flight, this becomes more cumbersome. To make up for this Selective Acknowledgements (SACK) were created. If in the handshake, both sides agree to use the SACK option, then instead of multiple DUP ACKs, the receiver can just send one packet with which specific packets were lost. Then just those packets are resent, and not the whole chain from the first lost packet. Whew, what a difference that can make.

Sequence/Segment numbering

It would be easier for us humans if packets were simply labeled A-Z or some other numbering convention. However, in the early days of little compute power, this did not make as much sense. Instead, packets were not simply "labeled", they used a method that matched the amount of data that was streaming across the wire. This is where sequence numbering comes in. If you have hit this point, make sure you are alert. This will take a little thinking.

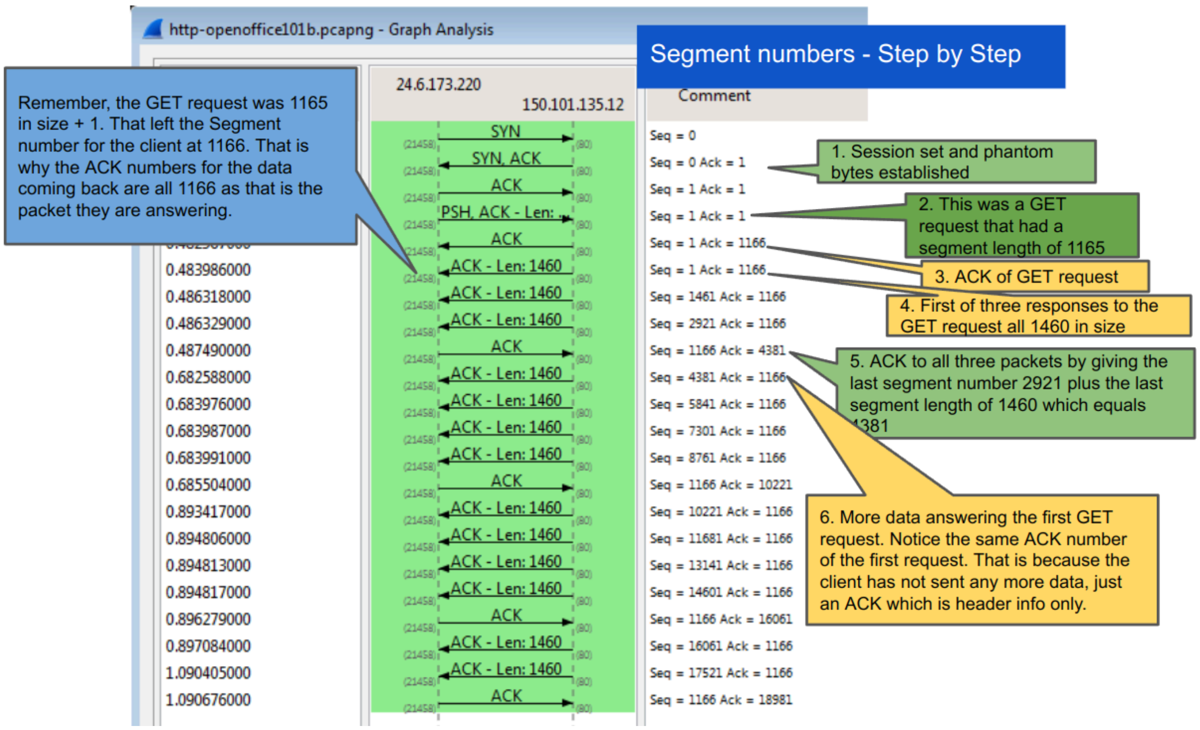
The way TCP keeps track of packets is by the measurement of the **segment** information it sends. What that means is, it measures the **ACTUAL** data you are sending, not the header information. The key thing to think about there is that the only time the sequence number increases is when there is data. ACK packets are just a flag bit set in the header. As such, it will not change the segment counter. Let's walk through this.



After the three way handshake, both sides set what is called the phantom byte. Both counters are set to one. One becomes the Sequence number of packet A. Packet A has a segment of 1460. Note that I wrote the SEGMENT size of 1460. That is not the overall packet length (which includes the header), that is simply the size of the actual data block that we are trying to get from point A to point B. The next packet then will get a sequence/segment number of the packet before it which was 1, plus the segment length 1460. So the sequence number for packet B is 1461. The next packet which is C is the number of the previous packet plus its segment size. I chose a uniform size so that makes it 1461+ the length of 1460, to then name packet C as 2921. Packet D is then 2921 + the length of 1460 to name packet D as 4381. Packet E is 4381 + 1460 to become 5841.

Previous packet number + segment length = packet number of the current packet.

Now here is where it gets interesting. Think of this from the receiver perspective. As the data streams in, it does not know how many packets were sent, because they did not need to be a uniform size. One of the above packets could just have easily been two smaller packets in size. All the receiver has to keep track of is gaps, and the size of those gaps, in the data coming in. That is how it asks for dropped packets with either DUP ACKs or SACKs. It does so by sequence number, or the range between sequence numbers missing. This allows the receiver to basically say: "Hey, I am missing this block of data!"
NOTE: This is important to realize. Please notice that sequence numbers only increase in one direction. That is the only way it can work. Data traveling from point A to B will constantly increase in one direction. The data that travels from B to A will have a different numbering. So when you look at sequence numbering be aware of that or you can get confused really really quickly.



Okay, are you ready? Let's walk through this step by step. Please note that it can take a few reads of this to get it in your head. I am going to attempt to explain it well enough.

I am going to label the side on the left Point A and the side on the right Point B.
It starts with the three way handshake. That sets all the relative sequence numbers to one in both directions.
Then from point A to point B a packet with the sequence number of 1 sends a GET request going to B. It is 1165 in length. That means that the counter from A to B is set to 1166.
The next packet is an ACK of the GET request sent from point B to point A. It is an ACK packet so it has zero segment length. Remember, only segment length counts, no data, no increase.
Then the data in answer to the GET request streams in from point B to A:
First packet is 1 with a length of 1460.
Second packet is 1461 with a length of 1460
Third packet is 2921 with a length of 1460.
Point A then sends an ACK to B of 4381.

That was three packets. One ACK can acknowledge multiple packets if they come in fast enough. By the ACK saying it acknowledged 4381 it acknowledged the length of all three. The sequence number it came from (Point A) was 1166. That is where we left off after the original GET request. It is an ACK so it does not have any segment data, so the numbering from A to B remains 1166.
Then Point B sends more data to A still answering the original GET request.
Packet 4 is 4381 with a length of 1460
Packet 5 is 5841 with a length of 1460
Packet 6 is 7301 with a length of 1460

Packet 7 is 8761 with a length of 1460
Point A then sends an ACK labeled 10221 from 1166.

Point A has not sent any new data, just acknowledgements, therefore the number does not change. Point A was able to acknowledge all 4 packets with just one acknowledgement that represents the summation of all the additional data.

Hopefully, you can see how the numbers are separate in each direction, and how you only increment when data is actually sent.

Take a breath, that was a lot to keep track of. I hate to do this, but you should know that the numbers don't actually start with one. The numbers you saw in the graph were what are known as relative numbers, and when you use the default settings in Wireshark, it will label them accordingly. However, when you look at the dissectors (comes up in Wireshark class in chapter 0) they also have a raw sequence number.

```
Transmission Control Protocol, Src Port: http (80), Dst Port: 35529 (35529), Seq: 8761, Ack: 348, Len: 1460
    Source Port: http (80)
    Destination Port: 35529 (35529)
    [Stream index: 11]
    [Conversation completeness: Complete, WITH_DATA (31)]
    [TCP Segment Len: 1460]
    Sequence Number: 8761    (relative sequence number)
    Sequence Number (raw): 3170740899
    [Next Sequence Number: 10221    (relative sequence number)]
    Acknowledgment Number: 348    (relative ack number)
    Acknowledgment number (raw): 3301962648
    0101 .... = Header Length: 20 bytes (5)
  > Flags: 0x010 (ACK)
    Window: 12501
```

The reason for this is that if the starting number of a sequence is always the same, that opens up the possibility for all sorts of easy manipulation. Sequence numbers then start with a random 32 bit number. Once you run out of sequence numbers they start once again at zero. The 32 bits set aside for sequence numbering in the protocol rolls over like an odometer to start the process again for as long as it needs.

Hope this helped

Please let me say one more thing.

If you made it this far, thanks for your time. Like I said at the beginning, this guide is not a replacement for Networking textbooks. The overall purpose for this guide is to present some of the standard material in a different context. Any topics or concepts that were unclear the first time around, hopefully, having a different view might have helped to clarify anything that was confusing or forgotten. Also, please recognize that learning networking, and IT in general, is a journey. It will never end, because the subject matter keeps developing and changing. The best way to really get to know it is to use it. If that is in a home lab, a lab on the job, or at the AZ cyber warfare range, getting hands on time is always the best way to learn. If you are taking the Wireshark class, get ready, the fun will continue. If you are in another class or using this simply as a reference, please continue to explore. Let this just be a starting point.

Also, recognize that networks and protocols are built by communities. That means that you can contribute as well. There is so much to do in IT, that everyone can have a part to play. Reach out to professionals, email them, connect on social media, ask questions. You might be surprised at how approachable they are. You might also find that they are a lot like you.

The only difference is, …the mileage.

-Will McCullen